# The Mutt E-Mail Client

by Michael Elkins <*me@cs.hmc.edu*> version 1.5.9

"All mail clients suck. This one just sucks less." -me, circa 1995

## Contents

# 1 Introduction

**Mutt** is a small but very powerful text-based MIME mail client. Mutt is highly configurable, and is well suited to the mail power user with advanced features like key bindings, keyboard macros, mail threading, regular expression searches and a powerful pattern matching language for selecting groups of messages.

## 1.1 Mutt Home Page

*http://www.mutt.org/*

## 1.2 Mailing Lists

To subscribe to one of the following mailing lists, send a message with the word *subscribe* in the body to `list-name-`*request*`@mutt.org`.

- *mutt-announce@mutt.org* – low traffic list for announcements
- *mutt-users@mutt.org* – help, bug reports and feature requests
- *mutt-dev@mutt.org* – development mailing list

**Note:** all messages posted to *mutt-announce* are automatically forwarded to *mutt-users*, so you do not need to be subscribed to both lists.

### 1.3 Software Distribution Sites

- *ftp://ftp.mutt.org/mutt/*

For a list of mirror sites, please refer to *http://www.mutt.org/download.html* .

### 1.4 IRC

Visit channel *#mutt* on *irc.freenode.net (www.freenode.net)* to chat with other people interested in Mutt.

### 1.5 USENET

See the newsgroup *comp.mail.mutt* .

### 1.6 Copyright



## 2 Getting Started

This section is intended as a brief overview of how to use Mutt. There are many other features which are described elsewhere in the manual. There is even more information available in the Mutt FAQ and various web pages. See the *Mutt Page* for more details.

The keybindings described in this section are the defaults as distributed. Your local system administrator may have altered the defaults for your site. You can always type "?" in any menu to display the current bindings.

The first thing you need to do is invoke mutt, simply by typing mutt at the command line. There are various command-line options, see either the mutt man page or the 6.1 (reference).

## 2.1 Moving Around in Menus

Information is presented in menus, very similar to ELM. Here is a table showing the common keys used to navigate menus in Mutt.

```
j or Down       next-entry      move to the next entry
k or Up         previous-entry  move to the previous entry
z or PageDn     page-down       go to the next page
Z or PageUp     page-up         go to the previous page
= or Home       first-entry     jump to the first entry
* or End        last-entry      jump to the last entry
q               quit            exit the current menu
?               help            list all keybindings for the current menu
```

## 2.2 Editing Input Fields

Mutt has a builtin line editor which is used as the primary way to input textual data such as email addresses or filenames. The keys used to move around while editing are very similar to those of Emacs.

```
^A or <Home>    bol             move to the start of the line
^B or <Left>    backward-char   move back one char
Esc B           backward-word   move back one word
^D or <Delete>  delete-char     delete the char under the cursor
^E or <End>     eol             move to the end of the line
^F or <Right>   forward-char    move forward one char
Esc F           forward-word    move forward one word
<Tab>           complete        complete filename or alias
^T              complete-query  complete address with query
^K              kill-eol        delete to the end of the line
ESC d           kill-eow        delete to the end ot the word
^W              kill-word       kill the word in front of the cursor
^U              kill-line       delete entire line
^V              quote-char      quote the next typed key
<Up>            history-up      recall previous string from history
<Down>          history-down    recall next string from history
<BackSpace>     backspace       kill the char in front of the cursor
Esc u           upcase-word     convert word to upper case
Esc l           downcase-word   convert word to lower case
Esc c           capitalize-word capitalize the word
^G              n/a             abort
<Return>        n/a             finish editing
```

You can remap the *editor* functions using the 3.3 (bind) command. For example, to make the *Delete* key delete the character in front of the cursor rather than under, you could use

```
bind editor <delete> backspace
```

## 2.3 Reading Mail - The Index and Pager

Similar to many other mail clients, there are two modes in which mail is read in Mutt. The first is the index of messages in the mailbox, which is called the "index" in Mutt. The second mode is the display of the message contents. This is called the "pager."

The next few sections describe the functions provided in each of these modes.

### 2.3.1 The Message Index

```
c               change to a different mailbox
ESC c           change to a folder in read-only mode
C               copy the current message to another mailbox
ESC C           decode a message and copy it to a folder
ESC s           decode a message and save it to a folder
D               delete messages matching a pattern
d               delete the current message
F               mark as important
l               show messages matching a pattern
N               mark message as new
o               change the current sort method
O               reverse sort the mailbox
q               save changes and exit
s               save-message
T               tag messages matching a pattern
t               toggle the tag on a message
ESC t           toggle tag on entire message thread
U               undelete messages matching a pattern
u               undelete-message
v               view-attachments
x               abort changes and exit
<Return>        display-message
<Tab>           jump to the next new message
@               show the author's full e-mail address
$               save changes to mailbox
/               search
ESC /           search-reverse
^L              clear and redraw the screen
^T              untag messages matching a pattern
```

**Status Flags** In addition to who sent the message and the subject, a short summary of the disposition of each message is printed beside the message number. Zero or more of the following "flags" may appear, which mean:

**D**

message is deleted (is marked for deletion)

**d**

message have attachments marked for deletion

**K**

contains a PGP public key

**N**

message is new

**O**

message is old

**P**

message is PGP encrypted

**r**

message has been replied to

**S**

message is signed, and the signature is succesfully verified

**s**

message is signed

**!**

message is flagged

*****

message is tagged

Some of the status flags can be turned on or off using

- **set-flag** (default: w)
- **clear-flag** (default: W)

Furthermore, the following flags reflect who the message is addressed to. They can be customized with the 6.3.265 ($to_chars) variable.

+

message is to you and you only

**T**

message is to you, but also to or cc'ed to others

**C**

message is cc'ed to you

**F**

message is from you

**L**

message is sent to a subscribed mailing list

### 2.3.2 The Pager

By default, Mutt uses its builtin pager to display the body of messages. The pager is very similar to the Unix program *less* though not nearly as featureful.

```
<Return>        go down one line
<Space>         display the next page (or next message if at the end of a message)
-               go back to the previous page
n               search for next match
S               skip beyond quoted text
T               toggle display of quoted text
?               show keybindings
/               search for a regular expression (pattern)
ESC /           search backwards for a regular expression
\               toggle search pattern coloring
^               jump to the top of the message
```

In addition, many of the functions from the *index* are available in the pager, such as *delete-message* or *copy-message* (this is one advantage over using an external pager to view messages).

Also, the internal pager supports a couple other advanced features. For one, it will accept and translate the "standard" nroff sequences for bold and underline. These sequences are a series of either the letter, backspace (^H), the letter again for bold or the letter, backspace, "_" for denoting underline. Mutt will attempt to display these in bold and underline respectively if your terminal supports them. If not, you can use the bold and underline 3.7 (color) objects to specify a color or mono attribute for them.

Additionally, the internal pager supports the ANSI escape sequences for character attributes. Mutt translates them into the correct color and character settings. The sequences Mutt supports are:

```
ESC [ Ps;Ps;Ps;...;Ps m
where Ps =
0    All Attributes Off
1    Bold on
4    Underline on
5    Blink on
7    Reverse video on
3x   Foreground color is x
4x   Background color is x

Colors are
0    black
1    red
2    green
3    yellow
4    blue
5    magenta
6    cyan
7    white
```

Mutt uses these attributes for handling text/enriched messages, and they can also be used by an external 5.4 (autoview) script for highlighting purposes. **Note:** If you change the colors for your display, for example by changing the color associated with color2 for your xterm, then that color will be used instead of green.

### 2.3.3 Threaded Mode

When the mailbox is 6.3.247 (sorted) by *threads*, there are a few additional functions available in the *index* and *pager* modes.

```
^D      delete-thread           delete all messages in the current thread
^U      undelete-thread         undelete all messages in the current thread
^N      next-thread             jump to the start of the next thread
^P      previous-thread         jump to the start of the previous thread
^R      read-thread             mark the current thread as read
ESC d   delete-subthread        delete all messages in the current subthread
ESC u   undelete-subthread      undelete all messages in the current subthread
ESC n   next-subthread          jump to the start of the next subthread
ESC p   previous-subthread      jump to the start of the previous subthread
ESC r   read-subthread          mark the current subthread as read
ESC t   tag-thread              toggle the tag on the current thread
ESC v   collapse-thread         toggle collapse for the current thread
ESC V   collapse-all            toggle collapse for all threads
P       parent-message          jump to parent message in thread
```

**Note:** Collapsing a thread displays only the first message in the thread and hides the others. This is useful when threads contain so many messages that you can only see a handful of threads on the screen. See %M in 6.3.92 ($index_format). For example, you could use "%?M?(#%03M)&(%4l)?" in 6.3.92 ($index_format) to optionally display the number of hidden messages if the thread is collapsed.

See also: 6.3.257 ($strict_threads).

### 2.3.4 Miscellaneous Functions

**create-alias** (default: a)

Creates a new alias based upon the current message (or prompts for a new one). Once editing is complete, an 3.2 (alias) command is added to the file specified by the 6.3.3 ($alias_file) variable for future use. **Note:** Specifying an 6.3.3 ($alias_file) does not add the aliases specified there-in, you must also 3.26 (source) the file.

**check-traditional-pgp** (default: ESC P)

This function will search the current message for content signed or encrypted with PGP the "traditional" way, that is, without proper MIME tagging. Technically, this function will temporarily change the MIME content types of the body parts containing PGP data; this is similar to the 2.3.4 (edit-type) function's effect.

**display-toggle-weed** (default: h)

Toggles the weeding of message header fields specified by 3.8 (ignore) commands.

**edit** (default: e)

This command (available in the "index" and "pager") allows you to edit the raw current message as it's present in the mail folder. After you have finished editing, the changed message will be appended to the current folder, and the original message will be marked for deletion.

**edit-type**
(default: ^E on the attachment menu, and in the pager and index menus; ^T on the compose menu)

This command is used to temporarily edit an attachment's content type to fix, for instance, bogus character set parameters. When invoked from the index or from the pager, you'll have the opportunity to edit the top-level attachment's content type. On the 5.1.2 (attachment menu), you can change any attachment's content type. These changes are not persistent, and get lost upon changing folders.

Note that this command is also available on the 5.1.3 (compose menu). There, it's used to fine-tune the properties of attachments you are going to send.

**enter-command** (default: ":")

This command is used to execute any command you would normally put in a configuration file. A common use is to check the settings of variables, or in conjunction with 3.6 (macros) to change settings on the fly.

**extract-keys** (default: ^K)

This command extracts PGP public keys from the current or tagged message(s) and adds them to your PGP public key ring.

**forget-passphrase** (default: ^F)

This command wipes the passphrase(s) from memory. It is useful, if you misspelled the passphrase.

**list-reply** (default: L)

Reply to the current or tagged message(s) by extracting any addresses which match the regular expressions given by the 3.10 (lists or subscribe) commands, but also honor any `Mail-Followup-To` header(s) if the 6.3.73 ($honor_followup_to) configuration variable is set. Using this when replying to messages posted to mailing lists helps avoid duplicate copies being sent to the author of the message you are replying to.

**pipe-message** (default: |)

Asks for an external Unix command and pipes the current or tagged message(s) to it. The variables 6.3.193 ($pipe_decode), 6.3.192 ($pipe_split), 6.3.194 ($pipe_sep) and 6.3.274 ($wait_key) control the exact behaviour of this function.

**resend-message** (default: ESC e)

With resend-message, mutt takes the current message as a template for a new message. This function is best described as "recall from arbitrary folders". It can conveniently be used to forward MIME messages while preserving the original mail structure. Note that the amount of headers included here depends on the value of the 6.3.275 ($weed) variable.

This function is also available from the attachment menu. You can use this to easily resend a message which was included with a bounce message as a message/rfc822 body part.

**shell-escape** (default: !)

Asks for an external Unix command and executes it. The 6.3.274 ($wait_key) can be used to control whether Mutt will wait for a key to be pressed when the command returns (presumably to let the user read the output of the command), based on the return status of the named command.

**toggle-quoted** (default: T)

The *pager* uses the 6.3.215 ($quote_regexp) variable to detect quoted text when displaying the body of the message. This function toggles the display of the quoted material in the message. It is particularly useful when are interested in just the response and there is a large amount of quoted text in the way.

**skip-quoted** (default: S)

This function will go to the next line of non-quoted text which come after a line of quoted text in the internal pager.

## 2.4 Sending Mail

The following bindings are available in the *index* for sending messages.

```
m       compose         compose a new message
r       reply           reply to sender
g       group-reply     reply to all recipients
L       list-reply      reply to mailing list address
f       forward         forward message
b       bounce          bounce (remail) message
ESC k   mail-key        mail a PGP public key to someone
```

Bouncing a message sends the message as is to the recipient you specify. Forwarding a message allows you to add comments or modify the message you are forwarding. These items are discussed in greater detail in the next chapter 2.5 ("Forwarding and Bouncing Mail").

Mutt will then enter the *compose* menu and prompt you for the recipients to place on the "To:" header field. Next, it will ask you for the "Subject:" field for the message, providing a default if you are replying to or forwarding a message. See also 6.3.10 ($askcc), 6.3.9 ($askbcc), 6.3.15 ($autoedit), 6.3.19 ($bounce), and 6.3.50 ($fast_reply) for changing how Mutt asks these questions.

Mutt will then automatically start your 6.3.46 ($editor) on the message body. If the 6.3.45 ($edit_headers) variable is set, the headers will be at the top of the message in your editor. Any messages you are replying to will be added in sort order to the message, with appropriate 6.3.14 ($attribution), 6.3.91 ($indent_string) and 6.3.204 ($post_indent_string). When forwarding a message, if the 6.3.114 ($mime_forward) variable is unset, a copy of the forwarded message will be included. If you have specified a 6.3.242 ($signature), it will be appended to the message.

Once you have finished editing the body of your mail message, you are returned to the *compose* menu. The following options are available:

```
a       attach-file             attach a file
```

```
A       attach-message          attach message(s) to the message
ESC k   attach-key              attach a PGP public key
d       edit-description        edit description on attachment
D       detach-file             detach a file
t       edit-to                 edit the To field
ESC f   edit-from               edit the From field
r       edit-reply-to           edit the Reply-To field
c       edit-cc                 edit the Cc field
b       edit-bcc                edit the Bcc field
y       send-message            send the message
s       edit-subject            edit the Subject
S       smime-menu              select S/MIME options
f       edit-fcc                specify an ‘‘Fcc’’ mailbox
p       pgp-menu                select PGP options
P       postpone-message        postpone this message until later
q       quit                    quit (abort) sending the message
w       write-fcc               write the message to a folder
i       ispell                  check spelling (if available on your system)
^F      forget-passphrase       wipe passphrase(s) from memory
```

**Note:** The attach-message function will prompt you for a folder to attach messages from. You can now tag messages in that folder and they will be attached to the message you are sending. Note that certain operations like composing a new mail, replying, forwarding, etc. are not permitted when you are in that folder. The %r in 6.3.255 ($status_format) will change to a 'A' to indicate that you are in attach-message mode.

### 2.4.1 Editing the message header

When editing the header of your outgoing message, there are a couple of special features available.

If you specify
`Fcc:` *filename*
Mutt will pick up *filename* just as if you had used the *edit-fcc* function in the *compose* menu.

You can also attach files to your message by specifying
`Attach:` *filename* [ *description* ]
where *filename* is the file to attach and *description* is an optional string to use as the description of the attached file.

When replying to messages, if you remove the *In-Reply-To:* field from the header field, Mutt will not generate a *References:* field, which allows you to create a new message thread.

Also see 6.3.45 (edit_headers).

### 2.4.2 Using Mutt with PGP

If you want to use PGP, you can specify

`Pgp:` [ `E` | `S` | `S`<*id*> ]

"E" encrypts, "S" signs and "S<id>" signs with the given key, setting 6.3.147 ($pgp_sign_as) permanently.

If you have told mutt to PGP encrypt a message, it will guide you through a key selection process when you try to send the message. Mutt will not ask you any questions about keys which have a certified user ID matching one of the message recipients' mail addresses. However, there may be situations in which there are several keys, weakly certified user ID fields, or where no matching keys can be found.

In these cases, you are dropped into a menu with a list of keys from which you can select one. When you quit this menu, or mutt can't find any matching keys, you are prompted for a user ID. You can, as usually, abort this prompt using `^G`. When you do so, mutt will return to the compose screen.

Once you have successfully finished the key selection, the message will be encrypted using the selected public keys, and sent out.

Most fields of the entries in the key selection menu (see also 6.3.139 ($pgp_entry_format)) have obvious meanings. But some explanations on the capabilities, flags, and validity fields are in order.

The flags sequence (%f) will expand to one of the following flags:

```
R           The key has been revoked and can't be used.
X           The key is expired and can't be used.
d           You have marked the key as disabled.
c           There are unknown critical self-signature
            packets.
```

The capabilities field (%c) expands to a two-character sequence representing a key's capabilities. The first character gives the key's encryption capabilities: A minus sign (**-**) means that the key cannot be used for encryption. A dot (**.**) means that it's marked as a signature key in one of the user IDs, but may also be used for encryption. The letter **e** indicates that this key can be used for encryption.

The second character indicates the key's signing capabilities. Once again, a "**-**" implies "not for signing", "**.**" implies that the key is marked as an encryption key in one of the user-ids, and "**s**" denotes a key which can be used for signing.

Finally, the validity field (%t) indicates how well-certified a user-id is. A question mark (**?**) indicates undefined validity, a minus character (**-**) marks an untrusted association, a space character means a partially trusted association, and a plus character (+) indicates complete validity.

### 2.4.3 Sending anonymous messages via mixmaster.

You may also have configured mutt to co-operate with Mixmaster, an anonymous remailer. Mixmaster permits you to send your messages anonymously using a chain of remailers. Mixmaster support in mutt is for mixmaster version 2.04 (beta 45 appears to be the latest) and 2.03. It does not support earlier versions or the later so-called version 3 betas, of which the latest appears to be called 2.9b23.

To use it, you'll have to obey certain restrictions. Most important, you cannot use the `Cc` and `Bcc` headers. To tell Mutt to use mixmaster, you have to select a remailer chain, using the mix function on the compose menu.

The chain selection screen is divided into two parts. In the (larger) upper part, you get a list of remailers you may use. In the lower part, you see the currently selected chain of remailers.

You can navigate in the chain using the `chain-prev` and `chain-next` functions, which are by default bound to the left and right arrows and to the `h` and `l` keys (think vi keyboard bindings). To insert a remailer at the current chain position, use the `insert` function. To append a remailer behind the current chain position,

use `select-entry` or `append`. You can also delete entries from the chain, using the corresponding function. Finally, to abandon your changes, leave the menu, or `accept` them pressing (by default) the `Return` key.

Note that different remailers do have different capabilities, indicated in the %c entry of the remailer menu lines (see 6.3.117 ($mix_entry_format)). Most important is the "middleman" capability, indicated by a capital "M": This means that the remailer in question cannot be used as the final element of a chain, but will only forward messages to other mixmaster remailers. For details on the other capabilities, please have a look at the mixmaster documentation.

## 2.5 Forwarding and Bouncing Mail

Bouncing and forwarding let you send an existing message to recipients that you specify. Bouncing a message uses the 6.3.237 (sendmail) command to send a copy to alternative addresses as if they were the message's original recipients. Forwarding a message, on the other hand, allows you to modify the message before it is resent (for example, by adding your own comments).

The following keys are bound by default:

```
f       forward         forward message
b       bounce          bounce (remail) message
```

Forwarding can be done by including the original message in the new message's body (surrounded by indicating lines) or including it as a MIME attachment, depending on the value of the 6.3.114 ($mime_forward) variable. Decoding of attachments, like in the pager, can be controlled by the 6.3.57 ($forward_decode) and 6.3.115 ($mime_forward_decode) variables, respectively. The desired forwarding format may depend on the content, therefore *$mime_forward* is a quadoption which, for example, can be set to "ask-no".

The inclusion of headers is controlled by the current setting of the 6.3.275 ($weed) variable, unless 6.3.114 (mime_forward) is set.

Editing the message to forward follows the same procedure as sending or replying to a message does.

## 2.6 Postponing Mail

At times it is desirable to delay sending a message that you have already begun to compose. When the *postpone-message* function is used in the *compose* menu, the body of your message and attachments are stored in the mailbox specified by the 6.3.206 ($postponed) variable. This means that you can recall the message even if you exit Mutt and then restart it at a later time.

Once a message is postponed, there are several ways to resume it. From the command line you can use the "-p" option, or if you *compose* a new message from the *index* or *pager* you will be prompted if postponed messages exist. If multiple messages are currently postponed, the *postponed* menu will pop up and you can select which message you would like to resume.

**Note:** If you postpone a reply to a message, the reply setting of the message is only updated when you actually finish the message and send it. Also, you must be in the same folder with the message you replied to for the status of the message to be updated.

See also the 6.3.205 ($postpone) quad-option.

# 3 Configuration

While the default configuration (or "preferences") make Mutt usable right out of the box, it is often desirable to tailor Mutt to suit your own tastes. When Mutt is first invoked, it will attempt to read the "system" configuration file (defaults set by your local system administrator), unless the "-n" 6.1 (command line) option is specified. This file is typically `/usr/local/share/mutt/Muttrc` or `/etc/Muttrc`. Mutt will next look for a file named `.muttrc` in your home directory. If this file does not exist and your home directory has a subdirectory named `.mutt`, mutt try to load a file named `.mutt/muttrc`.

`.muttrc` is the file where you will usually place your 6.2 (commands) to configure Mutt.

In addition, mutt supports version specific configuration files that are parsed instead of the default files as explained above. For instance, if your system has a `Muttrc-0.88` file in the system configuration directory, and you are running version 0.88 of mutt, this file will be sourced instead of the `Muttrc` file. The same is true of the user configuration file, if you have a file `.muttrc-0.88.6` in your home directory, when you run mutt version 0.88.6, it will source this file instead of the default `.muttrc` file. The version number is the same which is visible using the "-v" 6.1 (command line) switch or using the `show-version` key (default: V) from the index menu.

## 3.1 Syntax of Initialization Files

An initialization file consists of a series of 6.2 (commands). Each line of the file may contain one or more commands. When multiple commands are used, they must be separated by a semicolon (;).

```
set realname='Mutt user' ; ignore x-
```

The hash mark, or pound sign ("#"), is used as a "comment" character. You can use it to annotate your initialization file. All text after the comment character to the end of the line is ignored. For example,

```
my_hdr X-Disclaimer: Why are you listening to me? # This is a comment
```

Single quotes (') and double quotes (") can be used to quote strings which contain spaces or other special characters. The difference between the two types of quotes is similar to that of many popular shell programs, namely that a single quote is used to specify a literal string (one that is not interpreted for shell variables or quoting with a backslash [see next paragraph]), while double quotes indicate a string for which should be evaluated. For example, backtics are evaluated inside of double quotes, but **not** for single quotes.

\ quotes the next character, just as in shells such as bash and zsh. For example, if want to put quotes "" inside of a string, you can use "\" to force the next character to be a literal instead of interpreted character.

```
set realname="Michael \"MuttDude\" Elkins"
```

"\\" means to insert a literal "\" into the line. "\n" and "\r" have their usual C meanings of linefeed and carriage-return, respectively.

A \ at the end of a line can be used to split commands over multiple lines, provided that the split points don't appear in the middle of command names.

It is also possible to substitute the output of a Unix command in an initialization file. This is accomplished by enclosing the command in backquotes (`). For example,

```
my_hdr X-Operating-System: `uname -a`
```

The output of the Unix command "uname -a" will be substituted before the line is parsed. Note that since initialization files are line oriented, only the first line of output from the Unix command will be substituted.

UNIX environments can be accessed like the way it is done in shells like sh and bash: Prepend the name of the environment by a "$". For example,

```
set record=+sent_on_$HOSTNAME
```

The commands understood by mutt are explained in the next paragraphs. For a complete list, see the 6.2 (command reference).

## 3.2 Defining/Using aliases

Usage: `alias` *key address* [ , *address*, ... ]

It's usually very cumbersome to remember or type out the address of someone you are communicating with. Mutt allows you to create "aliases" which map a short string to a full address.

**Note:** if you want to create an alias for a group (by specifying more than one address), you **must** separate the addresses with a comma (",").

To remove an alias or aliases ("*" means all aliases):

`unalias` [ * | *key* ... ]

```
alias muttdude me@cs.hmc.edu (Michael Elkins)
alias theguys manny, moe, jack
```

Unlike other mailers, Mutt doesn't require aliases to be defined in a special file. The `alias` command can appear anywhere in a configuration file, as long as this file is 3.26 (sourced). Consequently, you can have multiple alias files, or you can have all aliases defined in your muttrc.

On the other hand, the 2.3.4 (create-alias) function can use only one file, the one pointed to by the 6.3.3 ($alias_file) variable (which is `~/.muttrc` by default). This file is not special either, in the sense that Mutt will happily append aliases to any file, but in order for the new aliases to take effect you need to explicitly 3.26 (source) this file too.

For example:

```
source /usr/local/share/Mutt.aliases
source ~/.mail_aliases
set alias_file=~/.mail_aliases
```

To use aliases, you merely use the alias at any place in mutt where mutt prompts for addresses, such as the *To:* or *Cc:* prompt. You can also enter aliases in your editor at the appropriate headers if you have the 6.3.45 ($edit_headers) variable set.

In addition, at the various address prompts, you can use the tab character to expand a partial alias to the full alias. If there are multiple matches, mutt will bring up a menu with the matching aliases. In order to

be presented with the full list of aliases, you must hit tab with out a partial alias, such as at the beginning of the prompt or after a comma denoting multiple addresses.

In the alias menu, you can select as many aliases as you want with the *select-entry* key (default: RET), and use the *exit* key (default: q) to return to the address prompt.

## 3.3 Changing the default key bindings

Usage: `bind` *map key function*

This command allows you to change the default key bindings (operation invoked when pressing a key).

*map* specifies in which menu the binding belongs. Multiple maps may be specified by separating them with commas (no additional whitespace is allowed). The currently defined maps are:

**generic**

This is not a real menu, but is used as a fallback for all of the other menus except for the pager and editor modes. If a key is not defined in another menu, Mutt will look for a binding to use in this menu. This allows you to bind a key to a certain function in multiple menus instead of having multiple bind statements to accomplish the same task.

**alias**

The alias menu is the list of your personal aliases as defined in your muttrc. It is the mapping from a short alias name to the full email address(es) of the recipient(s).

**attach**

The attachment menu is used to access the attachments on received messages.

**browser**

The browser is used for both browsing the local directory structure, and for listing all of your incoming mailboxes.

**editor**

The editor is the line-based editor the user enters text data.

**index**

The index is the list of messages contained in a mailbox.

**compose**

The compose menu is the screen used when sending a new message.

**pager**

The pager is the mode used to display message/attachment data, and help listings.

**pgp**

The pgp menu is used to select the OpenPGP keys used for encrypting outgoing messages.

**postpone**

The postpone menu is similar to the index menu, except is used when recalling a message the user was composing, but saved until later.

*key* is the key (or key sequence) you wish to bind. To specify a control character, use the sequence *\Cx*, where *x* is the letter of the control character (for example, to specify control-A use "\Ca"). Note that the case of *x* as well as *\C* is ignored, so that *\CA*, *\Ca*, *\cA* and *\ca* are all equivalent. An alternative form is to specify the key as a three digit octal number prefixed with a "\" (for example *\177* is equivalent to *\c?*).

In addition, *key* may consist of:

```
\t              tab
<tab>           tab
\r              carriage return
\n              newline
\e              escape
<esc>           escape
<up>            up arrow
<down>          down arrow
<left>          left arrow
<right>         right arrow
<pageup>        Page Up
<pagedown>      Page Down
<backspace>     Backspace
<delete>        Delete
<insert>        Insert
<enter>         Enter
<return>        Return
<home>          Home
<end>           End
<space>         Space bar
<f1>            function key 1
<f10>           function key 10
```

*key* does not need to be enclosed in quotes unless it contains a space (" ").

*function* specifies which action to take when *key* is pressed. For a complete list of functions, see the 6.4 (reference). The special function `noop` unbinds the specified key sequence.

## 3.4 Defining aliases for character sets

Usage: `charset-hook` *alias charset*
Usage: `iconv-hook` *charset local-charset*

The `charset-hook` command defines an alias for a character set. This is useful to properly display messages which are tagged with a character set name not known to mutt.

The `iconv-hook` command defines a system-specific name for a character set. This is helpful when your systems character conversion library insists on using strange, system-specific names for character sets.

## 3.5 Setting variables based upon mailbox

Usage: `folder-hook` [!]*regexp command*

It is often desirable to change settings based on which mailbox you are reading. The folder-hook command provides a method by which you can execute any configuration command. *regexp* is a regular expression specifying in which mailboxes to execute *command* before loading. If a mailbox matches multiple folder-hook's, they are executed in the order given in the muttrc.

**Note:** if you use the "!" shortcut for 6.3.253 ($spoolfile) at the beginning of the pattern, you must place it inside of double or single quotes in order to distinguish it from the logical *not* operator for the expression.

Note that the settings are *not* restored when you leave the mailbox. For example, a command action to perform is to change the sorting method based upon the mailbox being read:

```
folder-hook mutt set sort=threads
```

However, the sorting method is not restored to its previous value when reading a different mailbox. To specify a *default* command, use the pattern ".":

```
folder-hook . set sort=date-sent
```

## 3.6 Keyboard macros

Usage: `macro` *menu key sequence* [ *description* ]

Macros are useful when you would like a single key to perform a series of actions. When you press *key* in menu *menu*, Mutt will behave as if you had typed *sequence*. So if you have a common sequence of commands you type, you can create a macro to execute those commands with a single key.

*menu* is the 3.3 (map) which the macro will be bound. Multiple maps may be specified by separating multiple menu arguments by commas. Whitespace may not be used in between the menu arguments and the commas separating them.

*key* and *sequence* are expanded by the same rules as the 3.3 (key bindings). There are some additions however. The first is that control characters in *sequence* can also be specified as *^x*. In order to get a caret ("^") you need to use ^^. Secondly, to specify a certain key such as *up* or to invoke a function directly, you can use the format <*key name*> and <*function name*>. For a listing of key names see the section on 3.3 (key bindings). Functions are listed in the 6.4 (function reference).

The advantage with using function names directly is that the macros will work regardless of the current key bindings, so they are not dependent on the user having particular key definitions. This makes them more robust and portable, and also facilitates defining of macros in files used by more than one user (eg. the system Muttrc).

Optionally you can specify a descriptive text after *sequence*, which is shown in the help screens.

**Note:** Macro definitions (if any) listed in the help screen(s), are silently truncated at the screen width, and are not wrapped.

## 3.7 Using color and mono video attributes

Usage: `color` *object foreground background* [ *regexp* ]
Usage: `color` index *foreground background pattern*

Usage: `uncolor` index *pattern* [ *pattern* ... ]

If your terminal supports color, you can spice up Mutt by creating your own color scheme. To define the color of an object (type of information), you must specify both a foreground color **and** a background color (it is not possible to only specify one or the other).

*object* can be one of:

- attachment
- body (match *regexp* in the body of messages)
- bold (hiliting bold patterns in the body of messages)
- error (error messages printed by Mutt)
- header (match *regexp* in the message header)
- hdrdefault (default color of the message header in the pager)
- index (match *pattern* in the message index)
- indicator (arrow or bar used to indicate the current item in a menu)
- markers (the "+" markers at the beginning of wrapped lines in the pager)
- message (informational messages)
- normal
- quoted (text matching 6.3.215 ($quote_regexp) in the body of a message)
- quoted1, quoted2, ..., quoted**N** (higher levels of quoting)
- search (hiliting of words in the pager)
- signature
- status (mode lines used to display info about the mailbox or message)
- tilde (the "~" used to pad blank lines in the pager)
- tree (thread tree drawn in the message index and attachment menu)
- underline (hiliting underlined patterns in the body of messages)

*foreground* and *background* can be one of the following:

- white
- black
- green
- magenta

- blue
- cyan
- yellow
- red
- default
- color*x*

*foreground* can optionally be prefixed with the keyword `bright` to make the foreground color boldfaced (e.g., `brightred`).

If your terminal supports it, the special keyword *default* can be used as a transparent color. The value *brightdefault* is also valid. If Mutt is linked against the *S-Lang* library, you also need to set the *COLORFGBG* environment variable to the default colors of your terminal for this to work; for example (for Bourne-like shells):

```
set COLORFGBG="green;black"
export COLORFGBG
```

**Note:** The *S-Lang* library requires you to use the *lightgray* and *brown* keywords instead of *white* and *yellow* when setting this variable.

**Note:** The uncolor command can be applied to the index object only. It removes entries from the list. You **must** specify the same pattern specified in the color command for it to be removed. The pattern "*" is a special token which means to clear the color index list of all entries.

Mutt also recognizes the keywords *color0*, *color1*, ..., *color***N-1** (**N** being the number of colors supported by your terminal). This is useful when you remap the colors for your display (for example by changing the color associated with *color2* for your xterm), since color names may then lose their normal meaning.

If your terminal does not support color, it is still possible change the video attributes through the use of the "mono" command:

Usage: `mono` <*object*> <*attribute*> [ *regexp* ]
Usage: `mono` index *attribute pattern*
Usage: `unmono` index *pattern* [ *pattern* ... ]

where *attribute* is one of the following:

- none
- bold
- underline
- reverse
- standout

## 3.8 Ignoring (weeding) unwanted message headers

Usage: `[un]ignore` *pattern* [ *pattern* ... ]

Messages often have many header fields added by automatic processing systems, or which may not seem useful to display on the screen. This command allows you to specify header fields which you don't normally want to see.

You do not need to specify the full header field name. For example, "ignore content-" will ignore all header fields that begin with the pattern "content-". "ignore *" will ignore all headers.

To remove a previously added token from the list, use the "unignore" command. The "unignore" command will make Mutt display headers with the given pattern. For example, if you do "ignore x-" it is possible to "unignore x-mailer".

"unignore *" will remove all tokens from the ignore list.

For example:

```
# Sven's draconian header weeding
ignore *
unignore from date subject to cc
unignore organization organisation x-mailer: x-newsreader: x-mailing-list:
unignore posted-to:
```

## 3.9 Alternative addresses

Usage: `[un]alternates` *regexp* [ *regexp* ... ]

With various functions, mutt will treat messages differently, depending on whether you sent them or whether you received them from someone else. For instance, when replying to a message that you sent to a different party, mutt will automatically suggest to send the response to the original message's recipients – responding to yourself won't make much sense in many cases. (See 6.3.223 ($reply_to).)

Many users receive e-mail under a number of different addresses. To fully use mutt's features here, the program must be able to recognize what e-mail addresses you receive mail under. That's the purpose of the `alternates` command: It takes a list of regular expressions, each of which can identify an address under which you receive e-mail.

The `unalternates` command can be used to write exceptions to `alternates` patterns. If an address matches something in an `alternates` command, but you nonetheless do not think it is from you, you can list a more precise pattern under an `unalternates` command.

To remove a regular expression from the `alternates` list, use the `unalternates` command with exactly the same *regexp*. Likewise, if the *regexp* for a `alternates` command matches an entry on the `unalternates` list, that `unalternates` entry will be removed. If the *regexp* for `unalternates` is "*", *all entries* on `alternates` will be removed.

## 3.10 Mailing lists

Usage: `[un]lists` *regexp* [ *regexp* ... ]
Usage: `[un]subscribe` *regexp* [ *regexp* ... ]

Mutt has a few nice features for 4.8 (handling mailing lists). In order to take advantage of them, you must specify which addresses belong to mailing lists, and which mailing lists you are subscribed to. Once you have done this, the 2.3.4 (list-reply) function will work for all known lists. Additionally, when you send a message to a subscribed list, mutt will add a Mail-Followup-To header to tell other users' mail user agents not to send copies of replies to your personal address. Note that the Mail-Followup-To header is a non-standard extension which is not supported by all mail user agents. Adding it is not bullet-proof against receiving personal CCs of list messages. Also note that the generation of the Mail-Followup-To header is controlled by the 6.3.55 ($followup_to) configuration variable.

More precisely, Mutt maintains lists of patterns for the addresses of known and subscribed mailing lists. Every subscribed mailing list is known. To mark a mailing list as known, use the "lists" command. To mark it as subscribed, use "subscribe".

You can use regular expressions with both commands. To mark all messages sent to a specific bug report's address on mutt's bug tracking system as list mail, for instance, you could say "subscribe [0-9]*@bugs.guug.de". Often, it's sufficient to just give a portion of the list's e-mail address.

Specify as much of the address as you need to to remove ambiguity. For example, if you've subscribed to the Mutt mailing list, you will receive mail addresssed to *mutt-users@mutt.org*. So, to tell Mutt that this is a mailing list, you could add "lists mutt-users" to your initialization file. To tell mutt that you are subscribed to it, add "subscribe mutt-users" to your initialization file instead. If you also happen to get mail from someone whose address is *mutt-users@example.com*, you could use "lists mutt-users@mutt\\.org" or "subscribe mutt-users@mutt\\.org" to match only mail from the actual list.

The "unlists" command is used to remove a token from the list of known and subscribed mailing-lists. Use "unlists *" to remove all tokens.

To remove a mailing list from the list of subscribed mailing lists, but keep it on the list of known mailing lists, use "unsubscribe".

## 3.11 Using Multiple spool mailboxes

Usage: `mbox-hook` [!]*pattern mailbox*

This command is used to move read messages from a specified mailbox to a different mailbox automatically when you quit or change folders. *pattern* is a regular expression specifying the mailbox to treat as a "spool" mailbox and *mailbox* specifies where mail should be saved when read.

Unlike some of the other *hook* commands, only the *first* matching pattern is used (it is not possible to save read mail in more than a single mailbox).

## 3.12 Defining mailboxes which receive mail

Usage: `[un]mailboxes` [!]*filename* [ *filename* ... ]

This command specifies folders which can receive mail and which will be checked for new messages. By default, the main menu status bar displays how many of these folders have new messages.

When changing folders, pressing *space* will cycle through folders with new mail.

Pressing TAB in the directory browser will bring up a menu showing the files specified by the `mailboxes` command, and indicate which contain new messages. Mutt will automatically enter this mode when invoked from the command line with the `-y` option.

The "unmailboxes" command is used to remove a token from the list of folders which receive mail. Use "unmailboxes *" to remove all tokens.

**Note:** new mail is detected by comparing the last modification time to the last access time. Utilities like `biff` or `frm` or any other program which accesses the mailbox might cause Mutt to never detect new mail for that mailbox if they do not properly reset the access time. Backup tools are another common reason for updated access times.

**Note:** the filenames in the `mailboxes` command are resolved when the command is executed, so if these names contain 4.7 (shortcut characters) (such as "=" and "!"), any variable definition that affect these characters (like 6.3.53 ($folder) and 6.3.253 ($spoolfile)) should be executed before the `mailboxes` command.

## 3.13 User defined headers

Usage:
`my_hdr` *string*
`unmy_hdr` *field* [ *field* ... ]

The "my_hdr" command allows you to create your own header fields which will be added to every message you send.

For example, if you would like to add an "Organization:" header field to all of your outgoing messages, you can put the command

```
my_hdr Organization: A Really Big Company, Anytown, USA
```

in your `.muttrc`.

**Note:** space characters are *not* allowed between the keyword and the colon (":"). The standard for electronic mail (RFC822) says that space is illegal there, so Mutt enforces the rule.

If you would like to add a header field to a single message, you should either set the 6.3.45 (edit_headers) variable, or use the *edit-headers* function (default: "E") in the send-menu so that you can edit the header of your message along with the body.

To remove user defined header fields, use the "unmy_hdr" command. You may specify an asterisk ("*") to remove all header fields, or the fields to remove. For example, to remove all "To" and "Cc" header fields, you could use:

```
unmy_hdr to cc
```

## 3.14 Defining the order of headers when viewing messages

Usage: `hdr_order` *header1 header2 header3*

With this command, you can specify an order in which mutt will attempt to present headers to you when viewing messages.

"unhdr_order *" will clear all previous headers from the order list, thus removing the header order effects set by the system-wide startup file.

```
hdr_order From Date: From: To: Cc: Subject:
```

## 3.15 Specify default save filename

Usage: `save-hook` [!]*pattern filename*

This command is used to override the default filename used when saving messages. *filename* will be used as the default filename if the message is *From:* an address matching *regexp* or if you are the author and the message is addressed *to:* something matching *regexp*.

See 4.4.1 (Message Matching in Hooks) for information on the exact format of *pattern*.

Examples:

```
save-hook me@(turing\\.)?cs\\.hmc\\.edu$ +elkins
save-hook aol\\.com$ +spam
```

Also see the 3.17 (fcc-save-hook) command.

## 3.16 Specify default Fcc: mailbox when composing

Usage: `fcc-hook` [!]*pattern mailbox*

This command is used to save outgoing mail in a mailbox other than 6.3.220 ($record). Mutt searches the initial list of message recipients for the first matching *regexp* and uses *mailbox* as the default Fcc: mailbox. If no match is found the message will be saved to 6.3.220 ($record) mailbox.

See 4.4.1 (Message Matching in Hooks) for information on the exact format of *pattern*.

Example: `fcc-hook [@.]aol\\.com$ +spammers`

The above will save a copy of all messages going to the aol.com domain to the '+spammers' mailbox by default. Also see the 3.17 (fcc-save-hook) command.

## 3.17 Specify default save filename and default Fcc: mailbox at once

Usage: `fcc-save-hook` [!]*pattern mailbox*

This command is a shortcut, equivalent to doing both a 3.16 (fcc-hook) and a 3.15 (save-hook) with its arguments.

## 3.18 Change settings based upon message recipients

Usage: `reply-hook` [!]*pattern command*
Usage: `send-hook` [!]*pattern command*
Usage: `send2-hook` [!]*pattern command*

These commands can be used to execute arbitrary configuration commands based upon recipients of the message. *pattern* is a regular expression matching the desired address. *command* is executed when *regexp* matches recipients of the message.

`reply-hook` is matched against the message you are *replying* **to**, instead of the message you are *sending*. `send-hook` is matched against all messages, both *new* and *replies*. **Note:** `reply-hook`s are matched **before** the `send-hook`, **regardless** of the order specified in the users's configuration file.

`send2-hook` is matched every time a message is changed, either by editing it, or by using the compose menu to change its recipients or subject. `send2-hook` is executed after `send-hook`, and can, e.g., be used to set parameters such as the 6.3.237 ($sendmail) variable depending on the message's sender address.

For each type of `send-hook` or `reply-hook`, when multiple matches occur, commands are executed in the order they are specified in the muttrc (for that type of hook).

See 4.4.1 (Message Matching in Hooks) for information on the exact format of *pattern*.

Example: `send-hook mutt "set mime_forward signature=''"`

Another typical use for this command is to change the values of the 6.3.14 ($attribution), 6.3.242 ($signature) and 6.3.95 ($locale) variables in order to change the language of the attributions and signatures based upon the recipients.

**Note:** the send-hook's are only executed ONCE after getting the initial list of recipients. Adding a recipient after replying or editing the message will NOT cause any send-hook to be executed. Also note that my_hdr commands which modify recipient headers, or the message's subject, don't have any effect on the current message when executed from a send-hook.

## 3.19 Change settings before formatting a message

Usage: `message-hook` [!]*pattern command*

This command can be used to execute arbitrary configuration commands before viewing or formatting a message based upon information about the message. *command* is executed if the *pattern* matches the message to be displayed. When multiple matches occur, commands are executed in the order they are specified in the muttrc.

See 4.4.1 (Message Matching in Hooks) for information on the exact format of *pattern*.

Example:

```
message-hook ~A 'set pager=builtin'
message-hook '~f freshmeat-news' 'set pager="less \"+/^  subject: .*\""'
```

## 3.20 Choosing the cryptographic key of the recipient

Usage: `crypt-hook` *pattern keyid*

When encrypting messages with PGP or OpenSSL, you may want to associate a certain key with a given e-mail address automatically, either because the recipient's public key can't be deduced from the destination address, or because, for some reasons, you need to override the key Mutt would normally use. The crypt-hook command provides a method by which you can specify the ID of the public key to be used when encrypting messages to a certain recipient.

The meaning of "key id" is to be taken broadly in this context: You can either put a numerical key ID here, an e-mail address, or even just a real name.

## 3.21 Adding key sequences to the keyboard buffer

Usage: `push` *string*

This command adds the named string to the keyboard buffer. The string may contain control characters, key names and function names like the sequence string in the 3.6 (macro) command. You may use it to automatically run a sequence of commands at startup, or when entering certain folders.

## 3.22 Executing functions

Usage: `exec` *function* [ *function* ... ]

This command can be used to execute any function. Functions are listed in the 6.4 (function reference). "exec function" is equivalent to "push <function>".

## 3.23 Message Scoring

Usage: `score` *pattern value*
Usage: `unscore` *pattern* [ *pattern* ... ]

The `score` commands adds *value* to a message's score if *pattern* matches it. *pattern* is a string in the format described in the 4.2 (patterns) section (note: For efficiency reasons, patterns which scan information not available in the index, such as `~b`, `~B` or `~h`, may not be used). *value* is a positive or negative integer. A message's final score is the sum total of all matching `score` entries. However, you may optionally prefix *value* with an equal sign (=) to cause evaluation to stop at a particular entry if there is a match. Negative final scores are rounded up to 0.

The `unscore` command removes score entries from the list. You **must** specify the same pattern specified in the `score` command for it to be removed. The pattern "*" is a special token which means to clear the list of all score entries.

## 3.24 Spam detection

Usage: `spam` *pattern format*
Usage: `nospam` *pattern*

Mutt has generalized support for external spam-scoring filters. By defining your spam patterns with the `spam` and `nospam` commands, you can *limit*, *search*, and *sort* your mail based on its spam attributes, as determined by the external filter. You also can display the spam attributes in your index display using the `%H` selector in the 6.3.92 ($index_format) variable. (Tip: try `%?H?[%H] ?` to display spam tags only when they are defined for a given message.)

Your first step is to define your external filter's spam patterns using the `spam` command. *pattern* should be a regular expression that matches a header in a mail message. If any message in the mailbox matches this regular expression, it will receive a "spam tag" or "spam attribute" (unless it also matches a `nospam` pattern – see below.) The appearance of this attribute is entirely up to you, and is governed by the *format* parameter.

*format* can be any static text, but it also can include back-references from the *pattern* expression. (A regular expression "back-reference" refers to a sub-expression contained within parentheses.) `%1` is replaced with the first back-reference in the regex, `%2` with the second, etc.

If you're using multiple spam filters, a message can have more than one spam-related header. You can define `spam` patterns for each filter you use. If a message matches two or more of these patterns, and the $spam_separator variable is set to a string, then the message's spam tag will consist of all the *format* strings joined together, with the value of $spam_separator separating them.

For example, suppose I use DCC, SpamAssassin, and PureMessage. I might define these spam settings:

```
spam "X-DCC-.*-Metrics:.*(....)=many"         "90+/DCC-%1"
spam "X-Spam-Status: Yes"                     "90+/SA"
spam "X-PerlMX-Spam: .*Probability=([0-9]+)%" "%1/PM"
set spam_separator=", "
```

If I then received a message that DCC registered with "many" hits under the "Fuz2" checksum, and that PureMessage registered with a 97% probability of being spam, that message's spam tag would read `90+/DCC-Fuz2, 97/PM`. (The four characters before "=many" in a DCC report indicate the checksum used – in this case, "Fuz2".)

If the $spam_separator variable is unset, then each spam pattern match supercedes the previous one. Instead of getting joined *format* strings, you'll get only the last one to match.

The spam tag is what will be displayed in the index when you use `%H` in the `$index_format` variable. It's also the string that the `~H` pattern-matching expression matches against for *search* and *limit* functions. And it's what sorting by spam attribute will use as a sort key.

That's a pretty complicated example, and most people's actual environments will have only one spam filter. The simpler your configuration, the more effective mutt can be, especially when it comes to sorting.

Generally, when you sort by spam tag, mutt will sort *lexically* – that is, by ordering strings alphnumerically. However, if a spam tag begins with a number, mutt will sort numerically first, and lexically only when two numbers are equal in value. (This is like UNIX's `sort -n`.) A message with no spam attributes at all – that is, one that didn't match *any* of your `spam` patterns – is sorted at lowest priority. Numbers are sorted next, beginning with 0 and ranging upward. Finally, non-numeric strings are sorted, with "a" taking lower priority than "z". Clearly, in general, sorting by spam tags is most effective when you can coerce your filter to give you a raw number. But in case you can't, mutt can still do something useful.

The `nospam` command can be used to write exceptions to `spam` patterns. If a header pattern matches something in a `spam` command, but you nonetheless do not want it to receive a spam tag, you can list a more precise pattern under a `nospam` command.

If the *pattern* given to `nospam` is exactly the same as the *pattern* on an existing `spam` list entry, the effect will be to remove the entry from the spam list, instead of adding an exception. Likewise, if the *pattern* for a `spam` command matches an entry on the `nospam` list, that `nospam` entry will be removed. If the *pattern* for `nospam` is "*", *all entries on both lists* will be removed. This might be the default action if you use `spam` and `nospam` in conjunction with a `folder-hook`.

You can have as many `spam` or `nospam` commands as you like. You can even do your own primitive spam detection within mutt – for example, if you consider all mail from `MAILER-DAEMON` to be spam, you can use a `spam` command like this:

```
spam "^From: .*MAILER-DAEMON"       "999"
```

## 3.25 Setting variables

Usage: `set` [no|inv]*variable*[=*value*] [ *variable* ... ]
Usage: `toggle` *variable* [*variable* ... ]
Usage: `unset` *variable* [*variable* ... ]
Usage: `reset` *variable* [*variable* ... ]

This command is used to set (and unset) 6.3 (configuration variables). There are four basic types of variables: boolean, number, string and quadoption. *boolean* variables can be *set* (true) or *unset* (false). *number* variables can be assigned a positive integer value.

*string* variables consist of any number of printable characters. *strings* must be enclosed in quotes if they contain spaces or tabs. You may also use the "C" escape sequences **n** and **t** for newline and tab, respectively.

*quadoption* variables are used to control whether or not to be prompted for certain actions, or to specify a default action. A value of *yes* will cause the action to be carried out automatically as if you had answered yes to the question. Similarly, a value of *no* will cause the the action to be carried out as if you had answered "no." A value of *ask-yes* will cause a prompt with a default answer of "yes" and *ask-no* will provide a default answer of "no."

Prefixing a variable with "no" will unset it. Example: `set noaskbcc`.

For *boolean* variables, you may optionally prefix the variable name with `inv` to toggle the value (on or off). This is useful when writing macros. Example: `set invsmart_wrap`.

The `toggle` command automatically prepends the `inv` prefix to all specified variables.

The `unset` command automatically prepends the `no` prefix to all specified variables.

Using the enter-command function in the *index* menu, you can query the value of a variable by prefixing the name of the variable with a question mark:

```
set ?allow_8bit
```

The question mark is actually only required for boolean and quadoption variables.

The `reset` command resets all given variables to the compile time defaults (hopefully mentioned in this manual). If you use the command `set` and prefix the variable with "&" this has the same behavior as the reset command.

With the `reset` command there exists the special variable "all", which allows you to reset all variables to their system defaults.

## 3.26 Reading initialization commands from another file

Usage: `source` *filename*

This command allows the inclusion of initialization commands from other files. For example, I place all of my aliases in `~/.mail_aliases` so that I can make my `~/.muttrc` readable and keep my aliases private.

If the filename begins with a tilde ("~"), it will be expanded to the path of your home directory.

If the filename ends with a vertical bar (|), then *filename* is considered to be an executable program from which to read input (eg. `source ~/bin/myscript|`).

### 3.27 Removing hooks

Usage: `unhook` [ * | *hook-type* ]

This command permits you to flush hooks you have previously defined. You can either remove all hooks by giving the "*" character as an argument, or you can remove all hooks of a specific type by saying something like `unhook send-hook`.

## 4 Advanced Usage

### 4.1 Regular Expressions

All string patterns in Mutt including those in more complex 4.2 (patterns) must be specified using regular expressions (regexp) in the "POSIX extended" syntax (which is more or less the syntax used by egrep and GNU awk). For your convenience, we have included below a brief description of this syntax.

The search is case sensitive if the pattern contains at least one upper case letter, and case insensitive otherwise. Note that "\" must be quoted if used for a regular expression in an initialization command: "\\".

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.

Note that the regular expression can be enclosed/delimited by either " or ' which is useful if the regular expression includes a white-space character. See 3.1 (Syntax of Initialization Files) for more information on " and ' delimiter processing. To match a literal " or ' you must preface it with \ (backslash).

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any metacharacter with special meaning may be quoted by preceding it with a backslash.

The period "." matches any single character. The caret "^" and the dollar sign "$" are metacharacters that respectively match the empty string at the beginning and end of a line.

A list of characters enclosed by "[" and "]" matches any single character in that list; if the first character of the list is a caret "^" then it matches any character **not** in the list. For example, the regular expression **[0123456789]** matches any single digit. A range of ASCII characters may be specified by giving the first and last characters, separated by a hyphen "-". Most metacharacters lose their special meaning inside lists. To include a literal "]" place it first in the list. Similarly, to include a literal "^" place it anywhere but first. Finally, to include a literal hyphen "-" place it last.

Certain named classes of characters are predefined. Character classes consist of "[:", a keyword denoting the class, and ":]". The following classes are defined by the POSIX standard:

**[:alnum:]**

Alphanumeric characters.

**[:alpha:]**

Alphabetic characters.

**[:blank:]**

Space or tab characters.

**[:cntrl:]**

Control characters.

**[:digit:]**

Numeric characters.

**[:graph:]**

Characters that are both printable and visible. (A space is printable, but not visible, while an "a" is both.)

**[:lower:]**

Lower-case alphabetic characters.

**[:print:]**

Printable characters (characters that are not control characters.)

**[:punct:]**

Punctuation characters (characters that are not letter, digits, control characters, or space characters).

**[:space:]**

Space characters (such as space, tab and formfeed, to name a few).

**[:upper:]**

Upper-case alphabetic characters.

**[:xdigit:]**

Characters that are hexadecimal digits.

A character class is only valid in a regular expression inside the brackets of a character list. Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket list. For example, **[[:digit:]]** is equivalent to **[0-9]**.

Two additional special sequences can appear in character lists. These apply to non-ASCII character sets, which can have single symbols (called collating elements) that are represented with more than one character, as well as several characters that are equivalent for collating or sorting purposes:

**Collating Symbols**

A collating symbol is a multi-character collating element enclosed in "[." and ".]". For example, if "ch" is a collating element, then **[[.ch.]]** is a regexp that matches this collating element, while **[ch]** is a regexp that matches either "c" or "h".

**Equivalence Classes**

An equivalence class is a locale-specific name for a list of characters that are equivalent. The name is enclosed in "[=" and "=]". For example, the name "e" might be used to represent all of "è" "é" and "e". In this case, **[[=e=]]** is a regexp that matches any of "è", "é" and "e".

A regular expression matching a single character may be followed by one of several repetition operators:

**?**

The preceding item is optional and matched at most once.

*****

The preceding item will be matched zero or more times.

+

The preceding item will be matched one or more times.

**{n}**

The preceding item is matched exactly *n* times.

**{n,}**

The preceding item is matched *n* or more times.

**{,m}**

The preceding item is matched at most *m* times.

**{n,m}**

The preceding item is matched at least *n* times, but no more than *m* times.

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated subexpressions.

Two regular expressions may be joined by the infix operator "|"; the resulting regular expression matches any string matching either subexpression.

Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole subexpression may be enclosed in parentheses to override these precedence rules.

**Note:** If you compile Mutt with the GNU *rx* package, the following operators may also be used in regular expressions:

**y**

Matches the empty string at either the beginning or the end of a word.

**B**

Matches the empty string within a word.

\\<

Matches the empty string at the beginning of a word.

\\>

Matches the empty string at the end of a word.

**w**

Matches any word-constituent character (letter, digit, or underscore).

**W**

Matches any character that is not word-constituent.

\\‘

Matches the empty string at the beginning of a buffer (string).

\\’

Matches the empty string at the end of a buffer.

Please note however that these operators are not defined by POSIX, so they may or may not be available in stock libraries on various systems.

## 4.2 Patterns

Many of Mutt’s commands allow you to specify a pattern to match (limit, tag-pattern, delete-pattern, etc.). There are several ways to select messages:

```
~A              all messages
~b EXPR         messages which contain EXPR in the message body
~B EXPR         messages which contain EXPR in the whole message
~c USER         messages carbon-copied to USER
~C EXPR         message is either to: or cc: EXPR
~D              deleted messages
~d [MIN]-[MAX]  messages with ‘‘date-sent’’ in a Date range
~E              expired messages
~e EXPR         message which contains EXPR in the ‘‘Sender’’ field
~F              flagged messages
~f USER         messages originating from USER
~g              cryptographically signed messages
~G              cryptographically encrypted messages
~H EXPR         messages with a spam attribute matching EXPR
~h EXPR         messages which contain EXPR in the message header
~k              message contains PGP key material
~i ID           message which match ID in the ‘‘Message-ID’’ field
~L EXPR         message is either originated or received by EXPR
~l              message is addressed to a known mailing list
~m [MIN]-[MAX]  message in the range MIN to MAX *)
~n [MIN]-[MAX]  messages with a score in the range MIN to MAX *)
~N              new messages
~O              old messages
~p              message is addressed to you (consults alternates)
~P              message is from you (consults alternates)
~Q              messages which have been replied to
~R              read messages
~r [MIN]-[MAX]  messages with ‘‘date-received’’ in a Date range
~S              superseded messages
~s SUBJECT      messages having SUBJECT in the ‘‘Subject’’ field.
~T              tagged messages
~t USER         messages addressed to USER
~U              unread messages
~v              message is part of a collapsed thread.
~V              cryptographically verified messages
```

```
~x EXPR          messages which contain EXPR in the 'References' field
~y EXPR          messages which contain EXPR in the 'X-Label' field
~z [MIN]-[MAX]   messages with a size in the range MIN to MAX *)
~=               duplicated messages (see $duplicate_threads)
~$               unreferenced messages (requires threaded view)
```

Where EXPR, USER, ID, and SUBJECT are 4.1 (regular expressions). Special attention has to be made when using regular expressions inside of patterns. Specifically, Mutt's parser for these patterns will strip one level of backslash (\), which is normally used for quoting. If it is your intention to use a backslash in the regular expression, you will need to use two backslashes instead (\\).

*) The forms <[MAX], >[MIN], [MIN]- and -[MAX] are allowed, too.

### 4.2.1 Pattern Modifier

Note that patterns matching 'lists' of addresses (notably c,C,p,P and t) match if there is at least one match in the whole list. If you want to make sure that all elements of that list match, you need to prefix your pattern with ^. This example matches all mails which only has recipients from Germany.

```
^~C \.de$
```

### 4.2.2 Complex Patterns

Logical AND is performed by specifying more than one criterion. For example:

```
~t mutt ~f elkins
```

would select messages which contain the word "mutt" in the list of recipients **and** that have the word "elkins" in the "From" header field.

Mutt also recognizes the following operators to create more complex search patterns:

- ! – logical NOT operator
- | – logical OR operator
- () – logical grouping operator

Here is an example illustrating a complex search pattern. This pattern will select all messages which do not contain "mutt" in the "To" or "Cc" field and which are from "elkins".

```
!(~t mutt|~c mutt) ~f elkins
```

Here is an example using white space in the regular expression (note the ' and " delimiters). For this to match, the mail's subject must match the "^Junk +From +Me$" and it must be from either "Jim +Somebody" or "Ed +SomeoneElse":

```
'~s "^Junk +From +Me$" ~f ("Jim +Somebody"|"Ed +SomeoneElse")'
```

Note that if a regular expression contains parenthesis, or a veritical bar ("|"), you **must** enclose the expression in double or single quotes since those characters are also used to separate different parts of Mutt's pattern language. For example,

```
~f "me@(mutt\.org|cs\.hmc\.edu)"
```

Without the quotes, the parenthesis wouldn't end. This would be seperated to two OR'd patterns: *~f me@(mutt\.org* and *cs\.hmc\.edu)*. They are never what you want.

### 4.2.3 Searching by Date

Mutt supports two types of dates, *absolute* and *relative*.

**Absolute**. Dates **must** be in DD/MM/YY format (month and year are optional, defaulting to the current month and year). An example of a valid range of dates is:

```
Limit to messages matching: ~d 20/1/95-31/10
```

If you omit the minimum (first) date, and just specify "-DD/MM/YY", all messages *before* the given date will be selected. If you omit the maximum (second) date, and specify "DD/MM/YY-", all messages *after* the given date will be selected. If you specify a single date with no dash ("-"), only messages sent on the given date will be selected.

**Error Margins**. You can add error margins to absolute dates. An error margin is a sign (+ or -), followed by a digit, followed by one of the following units:

```
y       years
m       months
w       weeks
d       days
```

As a special case, you can replace the sign by a "*" character, which is equivalent to giving identical plus and minus error margins.

Example: To select any messages two weeks around January 15, 2001, you'd use the following pattern:

```
Limit to messages matching: ~d 15/1/2001*2w
```

**Relative**. This type of date is relative to the current date, and may be specified as:

- >*offset* (messages older than *offset* units)
- <*offset* (messages newer than *offset* units)
- =*offset* (messages exactly *offset* units old)

*offset* is specified as a positive number with one of the following units:

```
y       years
m       months
w       weeks
d       days
```

Example: to select messages less than 1 month old, you would use

```
Limit to messages matching: ~d <1m
```

**Note:** all dates used when searching are relative to the **local** time zone, so unless you change the setting of your 6.3.92 ($index_format) to include a `%[...]` format, these are **not** the dates shown in the main index.

## 4.3 Using Tags

Sometimes it is desirable to perform an operation on a group of messages all at once rather than one at a time. An example might be to save messages to a mailing list to a separate folder, or to delete all messages with a given subject. To tag all messages matching a pattern, use the tag-pattern function, which is bound to "shift-T" by default. Or you can select individual messages by hand using the "tag-message" function, which is bound to "t" by default. See 4.2 (patterns) for Mutt's pattern matching syntax.

Once you have tagged the desired messages, you can use the "tag-prefix" operator, which is the ";" (semicolon) key by default. When the "tag-prefix" operator is used, the **next** operation will be applied to all tagged messages if that operation can be used in that manner. If the 6.3.16 ($auto_tag) variable is set, the next operation applies to the tagged messages automatically, without requiring the "tag-prefix".

In 3.6 (macros) or 3.21 (push) commands, you can use the "tag-prefix-cond" operator. If there are no tagged messages, mutt will "eat" the rest of the macro to abort it's execution. Mutt will stop "eating" the macro when it encounters the "end-cond" operator; after this operator the rest of the macro will be executed as normal.

## 4.4 Using Hooks

A *hook* is a concept borrowed from the EMACS editor which allows you to execute arbitrary commands before performing some operation. For example, you may wish to tailor your configuration based upon which mailbox you are reading, or to whom you are sending mail. In the Mutt world, a *hook* consists of a 4.1 (regular expression) or 4.2 (pattern) along with a configuration option/command. See

- 3.5 (folder-hook)
- 3.18 (send-hook)
- 3.19 (message-hook)
- 3.15 (save-hook)
- 3.11 (mbox-hook)
- 3.16 (fcc-hook)
- 3.17 (fcc-save-hook)

for specific details on each type of *hook* available.

**Note:** if a hook changes configuration settings, these changes remain effective until the end of the current mutt session. As this is generally not desired, a default hook needs to be added before all other hooks to restore configuration defaults. Here is an example with send-hook and the my_hdr directive:

```
send-hook . 'unmy_hdr From:'
send-hook ~C'^b@b\.b$' my_hdr from: c@c.c
```

### 4.4.1 Message Matching in Hooks

Hooks that act upon messages (`send-hook, save-hook, fcc-hook, message-hook`) are evaluated in a slightly different manner. For the other types of hooks, a 4.1 (regular expression) is sufficient. But in dealing with messages a finer grain of control is needed for matching since for different purposes you want to match different criteria.

Mutt allows the use of the 4.2 (search pattern) language for matching messages in hook commands. This works in exactly the same way as it would when *limiting* or *searching* the mailbox, except that you are restricted to those operators which match information mutt extracts from the header of the message (i.e. from, to, cc, date, subject, etc.).

For example, if you wanted to set your return address based upon sending mail to a specific address, you could do something like:

```
send-hook '~t ^me@cs\.hmc\.edu$' 'my_hdr From: Mutt User <user@host>'
```

which would execute the given command when sending mail to *me@cs.hmc.edu*.

However, it is not required that you write the pattern to match using the full searching language. You can still specify a simple *regular expression* like the other hooks, in which case Mutt will translate your pattern into the full language, using the translation specified by the 6.3.36 ($default_hook) variable. The pattern is translated at the time the hook is declared, so the value of 6.3.36 ($default_hook) that is in effect at that time will be used.

## 4.5 External Address Queries

Mutt supports connecting to external directory databases such as LDAP, ph/qi, bbdb, or NIS through a wrapper script which connects to mutt using a simple interface. Using the 6.3.213 ($query_command) variable, you specify the wrapper command to use. For example:

```
set query_command = "mutt_ldap_query.pl '%s'"
```

The wrapper script should accept the query on the command-line. It should return a one line message, then each matching response on a single line, each line containing a tab separated address then name then some other optional information. On error, or if there are no matching addresses, return a non-zero exit code and a one line error message.

An example multiple response output:

```
Searching database ... 20 entries ... 3 matching:
me@cs.hmc.edu           Michael Elkins  mutt dude
blong@fiction.net       Brandon Long    mutt and more
roessler@guug.de        Thomas Roessler mutt pgp
```

There are two mechanisms for accessing the query function of mutt. One is to do a query from the index menu using the query function (default: Q). This will prompt for a query, then bring up the query menu which will list the matching responses. From the query menu, you can select addresses to create aliases, or to mail. You can tag multiple addresses to mail, start a new query, or have a new query appended to the current responses.

The other mechanism for accessing the query function is for address completion, similar to the alias completion. In any prompt for address entry, you can use the complete-query function (default: ^T) to run a query based on the current address you have typed. Like aliases, mutt will look for what you have typed back to the last space or comma. If there is a single response for that query, mutt will expand the address in place. If there are multiple responses, mutt will activate the query menu. At the query menu, you can select one or more addresses to be added to the prompt.

## 4.6 Mailbox Formats

Mutt supports reading and writing of four different mailbox formats: mbox, MMDF, MH and Maildir. The mailbox type is autodetected, so there is no need to use a flag for different mailbox types. When creating new mailboxes, Mutt uses the default specified with the 6.3.104 ($mbox_type) variable.

**mbox**. This is the most widely used mailbox format for UNIX. All messages are stored in a single file. Each message has a line of the form:

```
From me@cs.hmc.edu Fri, 11 Apr 1997 11:44:56 PST
```

to denote the start of a new message (this is often referred to as the "From_" line).

**MMDF**. This is a variant of the *mbox* format. Each message is surrounded by lines containing "^A^A^A^A" (four control-A's).

**MH**. A radical departure from *mbox* and *MMDF*, a mailbox consists of a directory and each message is stored in a separate file. The filename indicates the message number (however, this is may not correspond to the message number Mutt displays). Deleted messages are renamed with a comma (,) prepended to the filename. **Note:** Mutt detects this type of mailbox by looking for either `.mh_sequences` or `.xmhcache` (needed to distinguish normal directories from MH mailboxes).

**Maildir**. The newest of the mailbox formats, used by the Qmail MTA (a replacement for sendmail). Similar to *MH*, except that it adds three subdirectories of the mailbox: *tmp*, *new* and *cur*. Filenames for the messages are chosen in such a way they are unique, even when two programs are writing the mailbox over NFS, which means that no file locking is needed.

## 4.7 Mailbox Shortcuts

There are a number of built in shortcuts which refer to specific mailboxes. These shortcuts can be used anywhere you are prompted for a file or mailbox path.

- ! – refers to your 6.3.253 ($spoolfile) (incoming) mailbox
- > – refers to your 6.3.103 ($mbox) file
- < – refers to your 6.3.220 ($record) file
- - or !! – refers to the file you've last visited
- ~ – refers to your home directory
- = or + – refers to your 6.3.53 ($folder) directory
- @*alias* – refers to the 3.15 (default save folder) as determined by the address of the alias

## 4.8 Handling Mailing Lists

Mutt has a few configuration options that make dealing with large amounts of mail easier. The first thing you must do is to let Mutt know what addresses you consider to be mailing lists (technically this does not have to be a mailing list, but that is what it is most often used for), and what lists you are subscribed to. This is accomplished through the use of the 3.10 (lists and subscribe) commands in your muttrc.

Now that Mutt knows what your mailing lists are, it can do several things, the first of which is the ability to show the name of a list through which you received a message (i.e., of a subscribed list) in the *index* menu display. This is useful to distinguish between personal and list mail in the same mailbox. In the 6.3.92 ($index_format) variable, the escape "%L" will return the string "To <list>" when "list" appears in the "To" field, and "Cc <list>" when it appears in the "Cc" field (otherwise it returns the name of the author).

Often times the "To" and "Cc" fields in mailing list messages tend to get quite large. Most people do not bother to remove the author of the message they are reply to from the list, resulting in two or more copies being sent to that person. The "list-reply" function, which by default is bound to "L" in the *index* menu and *pager*, helps reduce the clutter by only replying to the known mailing list addresses instead of all recipients (except as specified by `Mail-Followup-To`, see below).

Mutt also supports the `Mail-Followup-To` header. When you send a message to a list of recipients which includes one or several subscribed mailing lists, and if the 6.3.55 ($followup_to) option is set, mutt will generate a Mail-Followup-To header which contains all the recipients to whom you send this message, but not your address. This indicates that group-replies or list-replies (also known as "followups") to this message should only be sent to the original recipients of the message, and not separately to you - you'll receive your copy through one of the mailing lists you are subscribed to.

Conversely, when group-replying or list-replying to a message which has a `Mail-Followup-To` header, mutt will respect this header if the 6.3.73 ($honor_followup_to) configuration variable is set. Using list-reply will in this case also make sure that the reply goes to the mailing list, even if it's not specified in the list of recipients in the `Mail-Followup-To`.

Note that, when header editing is enabled, you can create a `Mail-Followup-To` header manually. Mutt will only auto-generate this header if it doesn't exist when you send the message.

The other method some mailing list admins use is to generate a "Reply-To" field which points back to the mailing list address rather than the author of the message. This can create problems when trying to reply directly to the author in private, since most mail clients will automatically reply to the address given in the "Reply-To" field. Mutt uses the 6.3.223 ($reply_to) variable to help decide which address to use. If set to *ask-yes* or *ask-no*, you will be prompted as to whether or not you would like to use the address given in the

"Reply-To" field, or reply directly to the address given in the "From" field. When set to *yes*, the "Reply-To" field will be used when present.

The "X-Label:" header field can be used to further identify mailing lists or list subject matter (or just to annotate messages individually). The 6.3.92 ($index_format) variable's "%y" and "%Y" escapes can be used to expand "X-Label:" fields in the index, and Mutt's pattern-matcher can match regular expressions to "X-Label:" fields with the " y" selector. "X-Label:" is not a standard message header field, but it can easily be inserted by procmail and other mail filtering agents.

Lastly, Mutt has the ability to 6.3.247 (sort) the mailbox into 2.3.3 (threads). A thread is a group of messages which all relate to the same subject. This is usually organized into a tree-like structure where a message and all of its replies are represented graphically. If you've ever used a threaded news client, this is the same concept. It makes dealing with large volume mailing lists easier because you can easily delete uninteresting threads and quickly find topics of value.

## 4.9 Delivery Status Notification (DSN) Support

RFC1894 defines a set of MIME content types for relaying information about the status of electronic mail messages. These can be thought of as "return receipts." Berkeley sendmail 8.8.x currently has some command line options in which the mail client can make requests as to what type of status messages should be returned.

To support this, there are two variables. 6.3.42 ($dsn_notify) is used to request receipts for different results (such as failed message, message delivered, etc.). 6.3.43 ($dsn_return) requests how much of your message should be returned with the receipt (headers or full message). Refer to the man page on sendmail for more details on DSN.

## 4.10 POP3 Support (OPTIONAL)

If Mutt was compiled with POP3 support (by running the *configure* script with the *–enable-pop* flag), it has the ability to work with mailboxes located on a remote POP3 server and fetch mail for local browsing.

You can access the remote POP3 mailbox by selecting the folder `pop://popserver/`.

You can select an alternative port by specifying it with the server, ie: `pop://popserver:port/`.

You can also specify different username for each folder, ie: `pop://username@popserver[:port]/`.

Polling for new mail is more expensive over POP3 than locally. For this reason the frequency at which Mutt will check for mail remotely can be controlled by the 6.3.197 ($pop_checkinterval) variable, which defaults to every 60 seconds.

If Mutt was compiled with SSL support (by running the *configure* script with the *–with-ssl* flag), connections to POP3 servers can be encrypted. This naturally requires that the server supports SSL encrypted connections. To access a folder with POP3/SSL, you should use pops: prefix, ie: `pops://[username@]popserver[:port]/`.

Another way to access your POP3 mail is the *fetch-mail* function (default: G). It allows to connect to 6.3.199 (pop_host), fetch all your new mail and place it in the local 6.3.253 (spoolfile). After this point, Mutt runs exactly as if the mail had always been local.

**Note:** If you only need to fetch all messages to local mailbox you should consider using a specialized program, such as *fetchmail*

## 4.11 IMAP Support (OPTIONAL)

If Mutt was compiled with IMAP support (by running the *configure* script with the *–enable-imap* flag), it has the ability to work with folders located on a remote IMAP server.

You can access the remote inbox by selecting the folder `imap://imapserver/INBOX`, where `imapserver` is the name of the IMAP server and `INBOX` is the special name for your spool mailbox on the IMAP server. If you want to access another mail folder at the IMAP server, you should use `imap://imapserver/path/to/folder` where `path/to/folder` is the path of the folder you want to access.

You can select an alternative port by specifying it with the server, ie: `imap://imapserver:port/INBOX`.

You can also specify different username for each folder, ie: `imap://username@imapserver[:port]/INBOX`.

If Mutt was compiled with SSL support (by running the *configure* script with the *–with-ssl* flag), connections to IMAP servers can be encrypted. This naturally requires that the server supports SSL encrypted connections. To access a folder with IMAP/SSL, you should use `imaps://[username@]imapserver[:port]/path/to/folder` as your folder path.

Pine-compatible notation is also supported, ie `{[username@]imapserver[:port][/ssl]}path/to/folder`

Note that not all servers use / as the hierarchy separator. Mutt should correctly notice which separator is being used by the server and convert paths accordingly.

When browsing folders on an IMAP server, you can toggle whether to look at only the folders you are subscribed to, or all folders with the *toggle-subscribed* command. See also the 6.3.82 ($imap_list_subscribed) variable.

Polling for new mail on an IMAP server can cause noticeable delays. So, you'll want to carefully tune the 6.3.96 ($mail_check) and 6.3.263 ($timeout) variables. Personally I use

```
set mail_check=90
set timeout=15
```

with relatively good results over my slow modem line.

Note that if you are using mbox as the mail store on UW servers prior to v12.250, the server has been reported to disconnect a client if another client selects the same folder.

### 4.11.1 The Folder Browser

As of version 1.2, mutt supports browsing mailboxes on an IMAP server. This is mostly the same as the local file browser, with the following differences:

- In lieu of file permissions, mutt displays the string "IMAP", possibly followed by the symbol "+", indicating that the entry contains both messages and subfolders. On Cyrus-like servers folders will often contain both messages and subfolders.
- For the case where an entry can contain both messages and subfolders, the selection key (bound to `enter` by default) will choose to descend into the subfolder view. If you wish to view the messages in that folder, you must use `view-file` instead (bound to `space` by default).
- You can create, delete and rename mailboxes with the `create-mailbox`, `delete-mailbox`, and `rename-mailbox` commands (default bindings: `C`, `d` and `r`, respectively). You may also `subscribe` and `unsubscribe` to mailboxes (normally these are bound to `s` and `u`, respectively).

### 4.11.2 Authentication

Mutt supports four authentication methods with IMAP servers: SASL, GSSAPI, CRAM-MD5, and LOGIN (there is a patch by Grant Edwards to add NTLM authentication for you poor exchange users out there, but it has yet to be integrated into the main tree). There is also support for the pseudo-protocol ANONYMOUS, which allows you to log in to a public IMAP server without having an account. To use ANONYMOUS, simply make your username blank or "anonymous".

SASL is a special super-authenticator, which selects among several protocols (including GSSAPI, CRAM-MD5, ANONYMOUS, and DIGEST-MD5) the most secure method available on your host and the server. Using some of these methods (including DIGEST-MD5 and possibly GSSAPI), your entire session will be encrypted and invisible to those teeming network snoops. It is the best option if you have it. To use it, you must have the Cyrus SASL library installed on your system and compile mutt with the *–with-sasl* flag.

Mutt will try whichever methods are compiled in and available on the server, in the following order: SASL, ANONYMOUS, GSSAPI, CRAM-MD5, LOGIN.

There are a few variables which control authentication:

- 6.3.87 ($imap_user) - controls the username under which you request authentication on the IMAP server, for all authenticators. This is overridden by an explicit username in the mailbox path (ie by using a mailbox name of the form {`user@host`}).
- 6.3.83 ($imap_pass) - a password which you may preset, used by all authentication methods where a password is needed.
- 6.3.76 ($imap_authenticators) - a colon-delimited list of IMAP authentication methods to try, in the order you wish to try them. If specified, this overrides mutt's default (attempt everything, in the order listed above).

## 4.12 Managing multiple IMAP/POP accounts (OPTIONAL)

If you happen to have accounts on multiple IMAP and/or POP servers, you may find managing all the authentication settings inconvenient and error-prone. The account-hook command may help. This hook works like folder-hook but is invoked whenever you access a remote mailbox (including inside the folder browser), not just when you open the mailbox.

Some examples:

```
account-hook . 'unset imap_user; unset imap_pass; unset tunnel'
account-hook imap://host1/ 'set imap_user=me1 imap_pass=foo'
account-hook imap://host2/ 'set tunnel="ssh host2 /usr/libexec/imapd"'
```

## 4.13 Start a WWW Browser on URLs (EXTERNAL)

If a message contains URLs (*unified resource locator* = address in the WWW space like *http://www.mutt.org/*), it is efficient to get a menu with all the URLs and start a WWW browser on one of them. This functionality is provided by the external urlview program which can be retrieved at *ftp://ftp.mutt.org/mutt/contrib/* and the configuration commands:

```
macro index \cb |urlview\n
macro pager \cb |urlview\n
```

# 5 Mutt's MIME Support

Quite a bit of effort has been made to make Mutt the premier text-mode MIME MUA. Every effort has been made to provide the functionality that the discerning MIME user requires, and the conformance to the standards wherever possible. When configuring Mutt for MIME, there are two extra types of configuration files which Mutt uses. One is the `mime.types` file, which contains the mapping of file extensions to IANA MIME types. The other is the `mailcap` file, which specifies the external commands to use for handling specific MIME types.

## 5.1 Using MIME in Mutt

There are three areas/menus in Mutt which deal with MIME, they are the pager (while viewing a message), the attachment menu and the compose menu.

### 5.1.1 Viewing MIME messages in the pager

When you select a message from the index and view it in the pager, Mutt decodes the message to a text representation. Mutt internally supports a number of MIME types, including `text/plain, text/enriched, message/rfc822, and message/news`. In addition, the export controlled version of Mutt recognizes a variety of PGP MIME types, including PGP/MIME and application/pgp.

Mutt will denote attachments with a couple lines describing them. These lines are of the form:

```
[-- Attachment #1: Description --]
[-- Type: text/plain, Encoding: 7bit, Size: 10000 --]
```

Where the `Description` is the description or filename given for the attachment, and the `Encoding` is one of `7bit/8bit/quoted-printable/base64/binary`.

If Mutt cannot deal with a MIME type, it will display a message like:

```
[-- image/gif is unsupported (use 'v' to view this part) --]
```

### 5.1.2 The Attachment Menu

The default binding for `view-attachments` is 'v', which displays the attachment menu for a message. The attachment menu displays a list of the attachments in a message. From the attachment menu, you can save, print, pipe, delete, and view attachments. You can apply these operations to a group of attachments at once, by tagging the attachments and by using the "tag-prefix" operator. You can also reply to the current message from this menu, and only the current attachment (or the attachments tagged) will be quoted in your reply. You can view attachments as text, or view them using the mailcap viewer definition.

Finally, you can apply the usual message-related functions (like 2.3.4 (resend-message), and the reply and forward functions) to attachments of type `message/rfc822`.

See the help on the attachment menu for more information.

### 5.1.3 The Compose Menu

The compose menu is the menu you see before you send a message. It allows you to edit the recipient list, the subject, and other aspects of your message. It also contains a list of the attachments of your message, including the main body. From this menu, you can print, copy, filter, pipe, edit, compose, review, and rename an attachment or a list of tagged attachments. You can also modifying the attachment information, notably the type, encoding and description.

Attachments appear as follows:

```
-   1 [text/plain, 7bit, 1K]              /tmp/mutt-euler-8082-0 <no description>
    2 [applica/x-gunzip, base64, 422K]    ~/src/mutt-0.85.tar.gz <no description>
```

The '-' denotes that Mutt will delete the file after sending (or postponing, or cancelling) the message. It can be toggled with the `toggle-unlink` command (default: u). The next field is the MIME content-type, and can be changed with the `edit-type` command (default: ^T). The next field is the encoding for the attachment, which allows a binary message to be encoded for transmission on 7bit links. It can be changed with the `edit-encoding` command (default: ^E). The next field is the size of the attachment, rounded to kilobytes or megabytes. The next field is the filename, which can be changed with the `rename-file` command (default: R). The final field is the description of the attachment, and can be changed with the `edit-description` command (default: d).

## 5.2 MIME Type configuration with `mime.types`

When you add an attachment to your mail message, Mutt searches your personal mime.types file at `${HOME}/.mime.types`, and then the system mime.types file at `/usr/local/share/mutt/mime.types` or `/etc/mime.types`

The mime.types file consist of lines containing a MIME type and a space separated list of extensions. For example:

```
application/postscript          ps eps
application/pgp                 pgp
audio/x-aiff                    aif aifc aiff
```

A sample `mime.types` file comes with the Mutt distribution, and should contain most of the MIME types you are likely to use.

If Mutt can not determine the mime type by the extension of the file you attach, it will look at the file. If the file is free of binary information, Mutt will assume that the file is plain text, and mark it as `text/plain`. If the file contains binary information, then Mutt will mark it as `application/octet-stream`. You can change the MIME type that Mutt assigns to an attachment by using the `edit-type` command from the compose menu (default: ^T). The MIME type is actually a major mime type followed by the sub-type, separated by a '/'. 6 major types: application, text, image, video, audio, and model have been approved after various internet discussions. Mutt recognises all of these if the appropriate entry is found in the mime.types file. It also recognises other major mime types, such as the chemical type that is widely used in the molecular modelling community to pass molecular data in various forms to various molecular viewers. Non-recognised mime types should only be used if the recipient of the message is likely to be expecting such attachments.

## 5.3 MIME Viewer configuration with `mailcap`

Mutt supports RFC 1524 MIME Configuration, in particular the Unix specific format specified in Appendix A of RFC 1524. This file format is commonly referred to as the mailcap format. Many MIME compliant programs utilize the mailcap format, allowing you to specify handling for all MIME types in one place for all programs. Programs known to use this format include Netscape, XMosaic, lynx and metamail.

In order to handle various MIME types that Mutt can not handle internally, Mutt parses a series of external configuration files to find an external handler. The default search string for these files is a colon delimited list set to

```
${HOME}/.mailcap:/usr/local/share/mutt/mailcap:/etc/mailcap:/etc/mailcap:/usr/etc/mailcap:/usr/local/etc/mailc
```

where `$HOME` is your home directory.

In particular, the metamail distribution will install a mailcap file, usually as `/usr/local/etc/mailcap`, which contains some baseline entries.

### 5.3.1 The Basics of the mailcap file

A mailcap file consists of a series of lines which are comments, blank, or definitions.

A comment line consists of a # character followed by anything you want.

A blank line is blank.

A definition line consists of a content type, a view command, and any number of optional fields. Each field of a definition line is divided by a semicolon ';' character.

The content type is specified in the MIME standard type/subtype method. For example, `text/plain, text/html, image/gif,` etc. In addition, the mailcap format includes two formats for wildcards, one using the special '*' subtype, the other is the implicit wild, where you only include the major type. For example, `image/*`, or `video,` will match all image types and video types, respectively.

The view command is a Unix command for viewing the type specified. There are two different types of commands supported. The default is to send the body of the MIME message to the command on stdin. You can change this behaviour by using %s as a parameter to your view command. This will cause Mutt to save the body of the MIME message to a temporary file, and then call the view command with the %s replaced by the name of the temporary file. In both cases, Mutt will turn over the terminal to the view program until the program quits, at which time Mutt will remove the temporary file if it exists.

So, in the simplest form, you can send a text/plain message to the external pager more on stdin:

```
text/plain; more
```

Or, you could send the message as a file:

```
text/plain; more %s
```

Perhaps you would like to use lynx to interactively view a text/html message:

```
text/html; lynx %s
```

In this case, lynx does not support viewing a file from stdin, so you must use the %s syntax. **Note:** *Some older versions of lynx contain a bug where they will check the mailcap file for a viewer for text/html. They will find the line which calls lynx, and run it. This causes lynx to continuously spawn itself to view the object.*

On the other hand, maybe you don't want to use lynx interactively, you just want to have it convert the text/html to text/plain, then you can use:

```
text/html; lynx -dump %s | more
```

Perhaps you wish to use lynx to view text/html files, and a pager on all other text formats, then you would use the following:

```
text/html; lynx %s
text/*; more
```

This is the simplest form of a mailcap file.

### 5.3.2 Secure use of mailcap

The interpretion of shell meta-characters embedded in MIME parameters can lead to security problems in general. Mutt tries to quote parameters in expansion of %s syntaxes properly, and avoids risky characters by substituting them, see the 6.3.98 (mailcap_sanitize) variable.

Although mutt's procedures to invoke programs with mailcap seem to be safe, there are other applications parsing mailcap, maybe taking less care of it. Therefore you should pay attention to the following rules:

*Keep the %-expandos away from shell quoting.* Don't quote them with single or double quotes. Mutt does this for you, the right way, as should any other program which interprets mailcap. Don't put them into backtick expansions. Be highly careful with eval statements, and avoid them if possible at all. Trying to fix broken behaviour with quotes introduces new leaks - there is no alternative to correct quoting in the first place.

If you have to use the %-expandos' values in context where you need quoting or backtick expansions, put that value into a shell variable and reference the shell variable where necessary, as in the following example (using `$charset` inside the backtick expansion is safe, since it is not itself subject to any further expansion):

```
text/test-mailcap-bug; cat %s; copiousoutput; test=charset=%{charset} \
        && test "`echo $charset | tr '[A-Z]' '[a-z]'`" != iso-8859-1
```

### 5.3.3 Advanced mailcap Usage

**Optional Fields** In addition to the required content-type and view command fields, you can add semicolon ';' separated fields to set flags and other options. Mutt recognizes the following optional fields:

**copiousoutput**

This flag tells Mutt that the command passes possibly large amounts of text on stdout. This causes Mutt to invoke a pager (either the internal pager or the external pager defined by the pager variable) on the output of the view command. Without this flag, Mutt assumes that the command is interactive. One could use this to replace the pipe to `more` in the `lynx -dump` example in the Basic section:

```
text/html; lynx -dump %s ; copiousoutput
```

This will cause lynx to format the text/html output as text/plain and Mutt will use your standard pager to display the results.

**needsterminal**

Mutt uses this flag when viewing attachments with 5.4 (autoview), in order to decide whether it should honor the setting of the 6.3.274 ($wait_key) variable or not. When an attachment is viewed using an interactive program, and the corresponding mailcap entry has a *needsterminal* flag, Mutt will use 6.3.274 ($wait_key) and the exit status of the program to decide if it will ask you to press a key after the external program has exited. In all other situations it will not prompt you for a key.

**compose=<command>**

This flag specifies the command to use to create a new attachment of a specific MIME type. Mutt supports this from the compose menu.

**composetyped=<command>**

This flag specifies the command to use to create a new attachment of a specific MIME type. This command differs from the compose command in that mutt will expect standard MIME headers on the data. This can be used to specify parameters, filename, description, etc. for a new attachment. Mutt supports this from the compose menu.

**print=<command>**

This flag specifies the command to use to print a specific MIME type. Mutt supports this from the attachment and compose menus.

**edit=<command>**

This flag specifies the command to use to edit a specific MIME type. Mutt supports this from the compose menu, and also uses it to compose new attachments. Mutt will default to the defined editor for text attachments.

**nametemplate=<template>**

This field specifies the format for the file denoted by %s in the command fields. Certain programs will require a certain file extension, for instance, to correctly view a file. For instance, lynx will only interpret a file as `text/html` if the file ends in `.html`. So, you would specify lynx as a `text/html` viewer with a line in the mailcap file like:

```
text/html; lynx %s; nametemplate=%s.html
```

**test=<command>**

This field specifies a command to run to test whether this mailcap entry should be used. The command is defined with the command expansion rules defined in the next section. If the command returns 0, then the test passed, and Mutt uses this entry. If the command returns non-zero, then the test failed, and Mutt continues searching for the right entry. **Note:** *the content-type must match before Mutt performs the test.* For example:

```
text/html; netscape -remote 'openURL(%s)' ; test=RunningX
text/html; lynx %s
```

In this example, Mutt will run the program RunningX which will return 0 if the X Window manager is running, and non-zero if it isn't. If RunningX returns 0, then Mutt will call netscape to display the text/html object. If RunningX doesn't return 0, then Mutt will go on to the next entry and use lynx to display the text/html object.

**Search Order** When searching for an entry in the mailcap file, Mutt will search for the most useful entry for its purpose. For instance, if you are attempting to print an `image/gif`, and you have the following entries in your mailcap file, Mutt will search for an entry with the print command:

```
image/*;        xv %s
image/gif;      ; print= anytopnm %s | pnmtops | lpr; \
                nametemplate=%s.gif
```

Mutt will skip the `image/*` entry and use the `image/gif` entry with the print command.

In addition, you can use this with 5.4 (Autoview) to denote two commands for viewing an attachment, one to be viewed automatically, the other to be viewed interactively from the attachment menu. In addition, you can then use the test feature to determine which viewer to use interactively depending on your environment.

```
text/html;      netscape -remote 'openURL(%s)' ; test=RunningX
text/html;      lynx %s; nametemplate=%s.html
text/html;      lynx -dump %s; nametemplate=%s.html; copiousoutput
```

For 5.4 (Autoview), Mutt will choose the third entry because of the copiousoutput tag. For interactive viewing, Mutt will run the program RunningX to determine if it should use the first entry. If the program returns non-zero, Mutt will use the second entry for interactive viewing.

**Command Expansion** The various commands defined in the mailcap files are passed to the `/bin/sh` shell using the system() function. Before the command is passed to `/bin/sh -c`, it is parsed to expand various special parameters with information from Mutt. The keywords Mutt expands are:

**%s**

As seen in the basic mailcap section, this variable is expanded to a filename specified by the calling program. This file contains the body of the message to view/print/edit or where the composing program should place the results of composition. In addition, the use of this keyword causes Mutt to not pass the body of the message to the view/print/edit program on stdin.

**%t**

Mutt will expand %t to the text representation of the content type of the message in the same form as the first parameter of the mailcap definition line, ie `text/html` or `image/gif`.

**%{<parameter>}**

Mutt will expand this to the value of the specified parameter from the Content-Type: line of the mail message. For instance, if Your mail message contains:

```
Content-Type: text/plain; charset=iso-8859-1
```

then Mutt will expand %{charset} to iso-8859-1. The default metamail mailcap file uses this feature to test the charset to spawn an xterm using the right charset to view the message.

\%

This will be replaced by a %

Mutt does not currently support the %F and %n keywords specified in RFC 1524. The main purpose of these parameters is for multipart messages, which is handled internally by Mutt.

### 5.3.4 Example mailcap files

This mailcap file is fairly simple and standard:

```
# I'm always running X :)
video/*;        xanim %s > /dev/null
image/*;        xv %s > /dev/null

# I'm always running netscape (if my computer had more memory, maybe)
text/html;      netscape -remote 'openURL(%s)'
```

This mailcap file shows quite a number of examples:

```
# Use xanim to view all videos   Xanim produces a header on startup,
# send that to /dev/null so I don't see it
video/*;        xanim %s > /dev/null

# Send html to a running netscape by remote
text/html;      netscape -remote 'openURL(%s)'; test=RunningNetscape

# If I'm not running netscape but I am running X, start netscape on the
# object
text/html;      netscape %s; test=RunningX

# Else use lynx to view it as text
text/html;      lynx %s

# This version would convert the text/html to text/plain
text/html;      lynx -dump %s; copiousoutput

# I use enscript to print text in two columns to a page
text/*;         more %s; print=enscript -2Gr %s

# Netscape adds a flag to tell itself to view jpegs internally
image/jpeg;xv %s; x-mozilla-flags=internal

# Use xv to view images if I'm running X
# In addition, this uses the \ to extend the line and set my editor
# for images
image/*;xv %s; test=RunningX; \
```

```
        edit=xpaint %s

# Convert images to text using the netpbm tools
image/*;  (anytopnm %s | pnmscale -xysize 80 46 | ppmtopgm | pgmtopbm |
pbmtoascii -1x2 ) 2>&1 ; copiousoutput

# Send excel spreadsheets to my NT box
application/ms-excel; open.pl %s
```

## 5.4 MIME Autoview

In addition to explicitly telling Mutt to view an attachment with the MIME viewer defined in the mailcap file, Mutt has support for automatically viewing MIME attachments while in the pager.

To work, you must define a viewer in the mailcap file which uses the `copiousoutput` option to denote that it is non-interactive. Usually, you also use the entry to convert the attachment to a text representation which you can view in the pager.

You then use the `auto_view` muttrc command to list the content-types that you wish to view automatically.

For instance, if you set auto_view to:

```
auto_view text/html application/x-gunzip application/postscript image/gif application/x-tar-gz
```

Mutt could use the following mailcap entries to automatically view attachments of these types.

```
text/html;      lynx -dump %s; copiousoutput; nametemplate=%s.html
image/*;        anytopnm %s | pnmscale -xsize 80 -ysize 50 | ppmtopgm | pgmtopbm | pbmtoascii ; copiousoutput
application/x-gunzip;   gzcat; copiousoutput
application/x-tar-gz; gunzip -c %s | tar -tf - ; copiousoutput
application/postscript; ps2ascii %s; copiousoutput
```

"unauto_view" can be used to remove previous entries from the autoview list. This can be used with message-hook to autoview messages based on size, etc. "unauto_view *" will remove all previous entries.

## 5.5 MIME Multipart/Alternative

Mutt has some heuristics for determining which attachment of a multipart/alternative type to display. First, mutt will check the alternative_order list to determine if one of the available types is preferred. The alternative_order list consists of a number of mimetypes in order, including support for implicit and explicit wildcards, for example:

```
alternative_order text/enriched text/plain text application/postscript image/*
```

Next, mutt will check if any of the types have a defined 5.4 (auto_view), and use that. Failing that, Mutt will look for any text type. As a last attempt, mutt will look for any type it knows how to handle.

To remove a MIME type from the `alternative_order` list, use the `unalternative_order` command.

### 5.6 MIME Lookup

Mutt's mime_lookup list specifies a list of mime-types that should not be treated according to their mailcap entry. This option is designed to deal with binary types such as application/octet-stream. When an attachment's mime-type is listed in mime_lookup, then the extension of the filename will be compared to the list of extensions in the mime.types file. The mime-type associated with this extension will then be used to process the attachment according to the rules in the mailcap file and according to any other configuration options (such as auto_view) specified. Common usage would be:

```
mime_lookup application/octet-stream application/X-Lotus-Manuscript
```

In addition, the unmime_lookup command may be used to disable this feature for any particular mime-type if it had been set, for example, in a global muttrc.

## 6 Reference

### 6.1 Command line options

Running `mutt` with no arguments will make Mutt attempt to read your spool mailbox. However, it is possible to read other mailboxes and to send messages from the command line as well.

```
-A      expand an alias
-a      attach a file to a message
-b      specify a blind carbon-copy (BCC) address
-c      specify a carbon-copy (Cc) address
-e      specify a config command to be run after initilization files are read
-f      specify a mailbox to load
-F      specify an alternate file to read initialization commands
-h      print help on command line options
-H      specify a draft file from which to read a header and body
-i      specify a file to include in a message composition
-m      specify a default mailbox type
-n      do not read the system Muttrc
-p      recall a postponed message
-Q      query a configuration variable
-R      open mailbox in read-only mode
-s      specify a subject (enclose in quotes if it contains spaces)
-v      show version number and compile-time definitions
-x      simulate the mailx(1) compose mode
-y      show a menu containing the files specified by the mailboxes command
-z      exit immediately if there are no messages in the mailbox
-Z      open the first folder with new message,exit immediately if none
```

To read messages in a mailbox

`mutt` [ -nz ] [ -F *muttrc* ] [ -m *type* ] [ -f *mailbox* ]

To compose a new message

`mutt` [ -n ] [ -F *muttrc* ] [ -a *file* ] [ -c *address* ] [ -i *filename* ] [ -s *subject* ] *address* [ *address ...* ]

Mutt also supports a "batch" mode to send prepared messages. Simply redirect input from the file you wish to send. For example,

```
mutt -s "data set for run #2" professor@bigschool.edu < ~/run2.dat
```

This command will send a message to "professor@bigschool.edu" with a subject of "data set for run #2". In the body of the message will be the contents of the file "~/run2.dat".

## 6.2 Configuration Commands

The following are the commands understood by mutt.

- 4.12 (account-hook) *pattern command*
- 3.2 (alias) *key address* [ , *address*, ... ]
- 3.2 (unalias) [ * | *key* ... ]
- 3.9 (alternates) *regexp* [ *regexp* ... ]
- 3.9 (unalternates) [ * | *regexp* ... ]
- 5.5 (alternative_order) *mimetype* [ *mimetype* ... ]
- 5.5 (unalternative_order) *mimetype* [ *mimetype* ... ]
- 5.4 (auto_view) *mimetype* [ *mimetype* ... ]
- 5.4 (unauto_view) *mimetype* [ *mimetype* ... ]
- 3.3 (bind) *map key function*
- 3.4 (charset-hook) *alias charset*
- 3.7 (color) *object foreground background* [ *regexp* ]
- 3.7 (uncolor) *index pattern* [ *pattern* ... ]
- 3.22 (exec) *function* [ *function* ... ]
- 3.16 (fcc-hook) *pattern mailbox*
- 3.17 (fcc-save-hook) *pattern mailbox*
- 3.5 (folder-hook) *pattern command*
- 3.14 (hdr_order) *header* [ *header* ... ]
- 3.14 (unhdr_order) *header* [ *header* ... ]
- 3.4 (iconv-hook) *charset local-charset*
- 3.8 (ignore) *pattern* [ *pattern* ... ]
- 3.8 (unignore) *pattern* [ *pattern* ... ]
- 3.10 (lists) *regexp* [ *regexp* ... ]

- 3.10 (unlists) *regexp* [ *regexp* ... ]
- 3.6 (macro) *menu key sequence* [ *description* ]
- 3.12 (mailboxes) *filename* [ *filename* ... ]
- 3.11 (mbox-hook) *pattern mailbox*
- 3.19 (message-hook) *pattern command*
- 5.6 (mime_lookup) *mimetype* [ *mimetype* ... ]
- 5.6 (unmime_lookup) *mimetype* [ *mimetype* ... ]
- 3.7 (mono) *object attribute* [ *regexp* ]
- 3.7 (unmono) *index pattern* [ *pattern* ... ]
- 3.13 (my_hdr) *string*
- 3.13 (unmy_hdr) *field* [ *field* ... ]
- 3.20 (crypt-hook) *pattern key-id*
- 3.21 (push) *string*
- 3.25 (reset) *variable* [*variable* ... ]
- 3.15 (save-hook) *regexp filename*
- 3.23 (score) *pattern value*
- 3.23 (unscore) *pattern* [ *pattern* ... ]
- 3.18 (send-hook) *regexp command*
- 3.18 (reply-hook) *regexp command*
- 3.25 (set) [no|inv]*variable*[=*value*] [ *variable* ... ]
- 3.25 (unset) *variable* [*variable* ... ]
- 3.26 (source) *filename*
- 3.24 (spam) *pattern format*
- 3.24 (nospam) *pattern*
- 3.10 (subscribe) *regexp* [ *regexp* ... ]
- 3.10 (unsubscribe) *regexp* [ *regexp* ... ]
- 3.25 (toggle) *variable* [*variable* ... ]
- 3.27 (unhook) *hook-type*

## 6.3 Configuration variables

### 6.3.1 abort_nosubject

Type: quadoption
Default: ask-yes

If set to *yes*, when composing messages and no subject is given at the subject prompt, composition will be aborted. If set to *no*, composing messages with no subject given at the subject prompt will never be aborted.

### 6.3.2 abort_unmodified

Type: quadoption
Default: yes

If set to *yes*, composition will automatically abort after editing the message body if no changes are made to the file (this check only happens after the *first* edit of the file). When set to *no*, composition will never be aborted.

### 6.3.3 alias_file

Type: path
Default: "~/.muttrc"

The default file in which to save aliases created by the "2.3.4 (create-alias)" function.

**Note:** Mutt will not automatically source this file; you must explicitly use the "3.26 (source)" command for it to be executed.

### 6.3.4 alias_format

Type: string
Default: "%4n %2f %t %-10a %r"

Specifies the format of the data displayed for the 'alias' menu. The following printf(3)-style sequences are available:

**%a**

alias name

**%f**

flags - currently, a "d" for an alias marked for deletion

**%n**

index number

**%r**

address which alias expands to

**%t**

character which indicates if the alias is tagged for inclusion

### 6.3.5 allow_8bit

Type: boolean
Default: yes

Controls whether 8-bit data is converted to 7-bit using either Quoted- Printable or Base64 encoding when sending mail.

### 6.3.6 allow_ansi

Type: boolean
Default: no

Controls whether ANSI color codes in messages (and color tags in rich text messages) are to be interpreted. Messages containing these codes are rare, but if this option is set, their text will be colored accordingly. Note that this may override your color choices, and even present a security problem, since a message could include a line like "[– PGP output follows ..." and give it the same color as your attachment color.

### 6.3.7 arrow_cursor

Type: boolean
Default: no

When set, an arrow ("->") will be used to indicate the current entry in menus instead of highlighting the whole line. On slow network or modem links this will make response faster because there is less that has to be redrawn on the screen when moving to the next or previous entries in the menu.

### 6.3.8 ascii_chars

Type: boolean
Default: no

If set, Mutt will use plain ASCII characters when displaying thread and attachment trees, instead of the default *ACS* characters.

### 6.3.9 askbcc

Type: boolean
Default: no

If set, Mutt will prompt you for blind-carbon-copy (Bcc) recipients before editing an outgoing message.

### 6.3.10 askcc

Type: boolean
Default: no

If set, Mutt will prompt you for carbon-copy (Cc) recipients before editing the body of an outgoing message.

### 6.3.11 attach_format

Type: string
Default: "%u%D%I %t%4n %T%.40d%> [%.7m/%.10M, %.6e%?C?, %C?, %s] "

This variable describes the format of the 'attachment' menu. The following printf-style sequences are understood:

**%C**

charset

**%c**

requires charset conversion (n or c)

**%D**

deleted flag

**%d**

description

**%e**

MIME content-transfer-encoding

**%f**

filename

**%I**

disposition (I=inline, A=attachment)

**%m**

major MIME type

**%M**

MIME subtype

**%n**

attachment number

**%s**

size

**%t**

tagged flag

**%T**

graphic tree characters

**%u**

unlink (=to delete) flag

**%>X**

right justify the rest of the string and pad with character "X"

**%|X**

pad to the end of the line with character "X"

#### 6.3.12 attach_sep

Type: string
Default: "\n"

The separator to add between attachments when operating (saving, printing, piping, etc) on a list of tagged attachments.

#### 6.3.13 attach_split

Type: boolean
Default: yes

If this variable is unset, when operating (saving, printing, piping, etc) on a list of tagged attachments, Mutt will concatenate the attachments and will operate on them as a single attachment. The "6.3.12 ($attach_sep)" separator is added after each attachment. When set, Mutt will operate on the attachments one by one.

#### 6.3.14 attribution

Type: string
Default: "On %d, %n wrote:"

This is the string that will precede a message which has been included in a reply. For a full listing of defined printf()-like sequences see the section on "6.3.92 ($index_format)".

#### 6.3.15 autoedit

Type: boolean
Default: no

When set along with "6.3.45 ($edit_headers)", Mutt will skip the initial send-menu and allow you to immediately begin editing the body of your message. The send-menu may still be accessed once you have finished editing the body of your message.

Also see "6.3.50 ($fast_reply)".

#### 6.3.16 auto_tag

Type: boolean
Default: no

When set, functions in the *index* menu which affect a message will be applied to all tagged messages (if there are any). When unset, you must first use the tag-prefix function (default: ";") to make the next function apply to all tagged messages.

#### 6.3.17 beep

Type: boolean
Default: yes

When this variable is set, mutt will beep when an error occurs.

#### 6.3.18 beep_new

Type: boolean
Default: no

When this variable is set, mutt will beep whenever it prints a message notifying you of new mail. This is independent of the setting of the "6.3.17 ($beep)" variable.

#### 6.3.19 bounce

Type: quadoption
Default: ask-yes

Controls whether you will be asked to confirm bouncing messages. If set to *yes* you don't get asked if you want to bounce a message. Setting this variable to *no* is not generally useful, and thus not recommended, because you are unable to bounce messages.

#### 6.3.20 bounce_delivered

Type: boolean
Default: yes

When this variable is set, mutt will include Delivered-To headers when bouncing messages. Postfix users may wish to unset this variable.

#### 6.3.21 charset

Type: string
Default: ""

Character set your terminal uses to display and enter textual data.

#### 6.3.22 check_new

Type: boolean
Default: yes

**Note:** this option only affects *maildir* and *MH* style mailboxes.

When *set*, Mutt will check for new mail delivered while the mailbox is open. Especially with MH mailboxes, this operation can take quite some time since it involves scanning the directory and checking each file to see if it has already been looked at. If *check_new* is *unset*, no check for new mail is performed while the mailbox is open.

#### 6.3.23 collapse_unread

Type: boolean
Default: yes

When *unset*, Mutt will not collapse a thread if it contains any unread messages.

#### 6.3.24 uncollapse_jump

Type: boolean
Default: no

When *set*, Mutt will jump to the next unread message, if any, when the current thread is *un*collapsed.

#### 6.3.25 compose_format

Type: string
Default: "– Mutt: Compose [Approx. msg size: %l Atts: %a]%>-"

Controls the format of the status line displayed in the *Compose* menu. This string is similar to "6.3.255 ($status_format)", but has its own set of printf()-like sequences:

**%a**

total number of attachments

**%h**

local hostname

**%l**

approximate size (in bytes) of the current message

**%v**

Mutt version string

See the text describing the "6.3.255 ($status_format)" option for more information on how to set "6.3.25 ($compose_format)".

#### 6.3.26 config_charset

Type: string
Default: ""

When defined, Mutt will recode commands in rc files from this encoding.

### 6.3.27 confirmappend

Type: boolean
Default: yes

When set, Mutt will prompt for confirmation when appending messages to an existing mailbox.

### 6.3.28 confirmcreate

Type: boolean
Default: yes

When set, Mutt will prompt for confirmation when saving messages to a mailbox which does not yet exist before creating it.

### 6.3.29 connect_timeout

Type: number
Default: 30

Causes Mutt to timeout a network connection (for IMAP or POP) after this many seconds if the connection is not able to be established. A negative value causes Mutt to wait indefinitely for the connection to succeed.

### 6.3.30 content_type

Type: string
Default: "text/plain"

Sets the default Content-Type for the body of newly composed messages.

### 6.3.31 copy

Type: quadoption
Default: yes

This variable controls whether or not copies of your outgoing messages will be saved for later references. Also see "6.3.220 ($record)", "6.3.231 ($save_name)", "6.3.56 ($force_name)" and "3.16 (fcc-hook)".

### 6.3.32 crypt_use_gpgme

Type: boolean
Default: no

This variable controls the use the GPGME enabled crypto backends. If it is set and Mutt was build with gpgme support, the gpgme code for S/MIME and PGP will be used instead of the classic code. Note, that you need to use this option in .muttrc as it won't have any effect when used interactively.

### 6.3.33 crypt_autopgp

Type: boolean
Default: yes

This variable controls whether or not mutt may automatically enable PGP encryption/signing for messages. See also "6.3.128 ($crypt_autoencrypt)", "6.3.130 ($crypt_replyencrypt)", "6.3.127 ($crypt_autosign)", "6.3.131 ($crypt_replysign)" and "6.3.136 ($smime_is_default)".

### 6.3.34 crypt_autosmime

Type: boolean
Default: yes

This variable controls whether or not mutt may automatically enable S/MIME encryption/signing for messages. See also "6.3.128 ($crypt_autoencrypt)", "6.3.130 ($crypt_replyencrypt)", "6.3.127 ($crypt_autosign)", "6.3.131 ($crypt_replysign)" and "6.3.136 ($smime_is_default)".

### 6.3.35 date_format

Type: string
Default: "!%a, %b %d, %Y at %I:%M:%S%p %Z"

This variable controls the format of the date printed by the "%d" sequence in "6.3.92 ($index_format)". This is passed to the *strftime* call to process the date. See the man page for *strftime(3)* for the proper syntax.

Unless the first character in the string is a bang ("!"), the month and week day names are expanded according to the locale specified in the variable "6.3.95 ($locale)". If the first character in the string is a bang, the bang is discarded, and the month and week day names in the rest of the string are expanded in the *C* locale (that is in US English).

### 6.3.36 default_hook

Type: string
Default: "~f %s !~P | (~P ~C %s)"

This variable controls how send-hooks, message-hooks, save-hooks, and fcc-hooks will be interpreted if they are specified with only a simple regexp, instead of a matching pattern. The hooks are expanded when they are declared, so a hook will be interpreted according to the value of this variable at the time the hook is declared. The default value matches if the message is either from a user matching the regular expression given, or if it is from you (if the from address matches "alternates") and is to or cc'ed to a user matching the given regular expression.

### 6.3.37 delete

Type: quadoption
Default: ask-yes

Controls whether or not messages are really deleted when closing or synchronizing a mailbox. If set to *yes*, messages marked for deleting will automatically be purged without prompting. If set to *no*, messages marked for deletion will be kept in the mailbox.

### 6.3.38 delete_untag

Type: boolean
Default: yes

If this option is *set*, mutt will untag messages when marking them for deletion. This applies when you either explicitly delete a message, or when you save it to another folder.

### 6.3.39 digest_collapse

Type: boolean
Default: yes

If this option is *set*, mutt's received-attachments menu will not show the subparts of individual messages in a multipart/digest. To see these subparts, press 'v' on that menu.

### 6.3.40 display_filter

Type: path
Default: ""

When set, specifies a command used to filter messages. When a message is viewed it is passed as standard input to 6.3.40 ($display_filter), and the filtered message is read from the standard output.

### 6.3.41 dotlock_program

Type: path
Default: "/usr/local/bin/mutt_dotlock"

Contains the path of the mutt_dotlock (8) binary to be used by mutt.

### 6.3.42 dsn_notify

Type: string
Default: ""

**Note:** you should not enable this unless you are using Sendmail 8.8.x or greater.

This variable sets the request for when notification is returned. The string consists of a comma separated list (no spaces!) of one or more of the following: *never*, to never request notification, *failure*, to request notification on transmission failure, *delay*, to be notified of message delays, *success*, to be notified of successful transmission.

Example: set dsn_notify="failure,delay"

#### 6.3.43 dsn_return

Type: string
Default: ""

**Note:** you should not enable this unless you are using Sendmail 8.8.x or greater.

This variable controls how much of your message is returned in DSN messages. It may be set to either *hdrs* to return just the message header, or *full* to return the full message.

Example: set dsn_return=hdrs

#### 6.3.44 duplicate_threads

Type: boolean
Default: yes

This variable controls whether mutt, when sorting by threads, threads messages with the same message-id together. If it is set, it will indicate that it thinks they are duplicates of each other with an equals sign in the thread diagram.

#### 6.3.45 edit_headers

Type: boolean
Default: no

This option allows you to edit the header of your outgoing messages along with the body of your message.

#### 6.3.46 editor

Type: path
Default: ""

This variable specifies which editor is used by mutt. It defaults to the value of the VISUAL, or EDITOR, environment variable, or to the string "vi" if neither of those are set.

#### 6.3.47 encode_from

Type: boolean
Default: no

When *set*, mutt will quoted-printable encode messages when they contain the string "From " in the beginning of a line. Useful to avoid the tampering certain mail delivery and transport agents tend to do with messages.

#### 6.3.48 envelope_from

Type: boolean
Default: no

When *set*, mutt will try to derive the message's *envelope* sender from the "From:" header. Note that this information is passed to sendmail command using the "-f" command line switch, so don't set this option if

you are using that switch in 6.3.237 ($sendmail) yourself, or if the sendmail on your machine doesn't support that command line switch.

### 6.3.49 escape

Type: string
Default: "~"

Escape character to use for functions in the builtin editor.

### 6.3.50 fast_reply

Type: boolean
Default: no

When set, the initial prompt for recipients and subject are skipped when replying to messages, and the initial prompt for subject is skipped when forwarding messages.

**Note:** this variable has no effect when the "6.3.15 ($autoedit)" variable is set.

### 6.3.51 fcc_attach

Type: boolean
Default: yes

This variable controls whether or not attachments on outgoing messages are saved along with the main body of your message.

### 6.3.52 fcc_clear

Type: boolean
Default: no

When this variable is set, FCCs will be stored unencrypted and unsigned, even when the actual message is encrypted and/or signed. (PGP only)

### 6.3.53 folder

Type: path
Default: "~/Mail"

Specifies the default location of your mailboxes. A '+' or '=' at the beginning of a pathname will be expanded to the value of this variable. Note that if you change this variable from the default value you need to make sure that the assignment occurs *before* you use '+' or '=' for any other variables since expansion takes place during the 'set' command.

### 6.3.54 folder_format

Type: string
Default: "%2C %t %N %F %2l %-8.8u %-8.8g %8s %d %f"

This variable allows you to customize the file browser display to your personal taste. This string is similar to "6.3.92 ($index_format)", but has its own set of printf()-like sequences:

**%C**

current file number

**%d**

date/time folder was last modified

**%f**

filename

**%F**

file permissions

**%g**

group name (or numeric gid, if missing)

**%l**

number of hard links

**%N**

N if folder has new mail, blank otherwise

**%s**

size in bytes

**%t**

* if the file is tagged, blank otherwise

**%u**

owner name (or numeric uid, if missing)

**%>X**

right justify the rest of the string and pad with character "X"

**%|X**

pad to the end of the line with character "X"

### 6.3.55 followup_to

Type: boolean
Default: yes

Controls whether or not the *Mail-Followup-To* header field is generated when sending mail. When *set*, Mutt will generate this field when you are replying to a known mailing list, specified with the "subscribe" or "3.10 (lists)" commands.

This field has two purposes. First, preventing you from receiving duplicate copies of replies to messages which you send to mailing lists, and second, ensuring that you do get a reply separately for any messages sent to known lists to which you are not subscribed. The header will contain only the list's address for subscribed lists, and both the list address and your own email address for unsubscribed lists. Without this header, a group reply to your message sent to a subscribed list will be sent to both the list and your address, resulting in two copies of the same email for you.

### 6.3.56 force_name

Type: boolean
Default: no

This variable is similar to "6.3.231 ($save_name)", except that Mutt will store a copy of your outgoing message by the username of the address you are sending to even if that mailbox does not exist.

Also see the "6.3.220 ($record)" variable.

### 6.3.57 forward_decode

Type: boolean
Default: yes

Controls the decoding of complex MIME messages into text/plain when forwarding a message. The message header is also RFC2047 decoded. This variable is only used, if "6.3.114 ($mime_forward)" is *unset*, otherwise "6.3.115 ($mime_forward_decode)" is used instead.

### 6.3.58 forward_edit

Type: quadoption
Default: yes

This quadoption controls whether or not the user is automatically placed in the editor when forwarding messages. For those who always want to forward with no modification, use a setting of "no".

### 6.3.59 forward_format

Type: string
Default: "[%a: %s]"

This variable controls the default subject when forwarding a message. It uses the same format sequences as the "6.3.92 ($index_format)" variable.

### 6.3.60 forward_quote

Type: boolean
Default: no

When *set* forwarded messages included in the main body of the message (when "6.3.114 ($mime_forward)" is *unset*) will be quoted using "6.3.91 ($indent_string)".

### 6.3.61 from

Type: e-mail address
Default: ""

When set, this variable contains a default from address. It can be overridden using my_hdr (including from send-hooks) and "6.3.226 ($reverse_name)". This variable is ignored if "6.3.269 ($use_from)" is unset.

Defaults to the contents of the environment variable EMAIL.

### 6.3.62 gecos_mask

Type: regular expression
Default: "^[^,]*"

A regular expression used by mutt to parse the GECOS field of a password entry when expanding the alias. By default the regular expression is set to "^[^,]*" which will return the string up to the first "," encountered. If the GECOS field contains a string like "lastname, firstname" then you should set the gecos_mask=".*".

This can be useful if you see the following behavior: you address a e-mail to user ID stevef whose full name is Steve Franklin. If mutt expands stevef to "Franklin" stevef@foo.bar then you should set the gecos_mask to a regular expression that will match the whole name so mutt will expand "Franklin" to "Franklin, Steve".

### 6.3.63 hdrs

Type: boolean
Default: yes

When unset, the header fields normally added by the "3.13 (my_hdr)" command are not created. This variable *must* be unset before composing a new message or replying in order to take effect. If set, the user defined header fields are added to every new message.

### 6.3.64 header

Type: boolean
Default: no

When set, this variable causes Mutt to include the header of the message you are replying to into the edit buffer. The "6.3.275 ($weed)" setting applies.

#### 6.3.65 help

Type: boolean
Default: yes

When set, help lines describing the bindings for the major functions provided by each menu are displayed on the first line of the screen.

**Note:** The binding will not be displayed correctly if the function is bound to a sequence rather than a single keystroke. Also, the help line may not be updated if a binding is changed while Mutt is running. Since this variable is primarily aimed at new users, neither of these should present a major problem.

#### 6.3.66 hidden_host

Type: boolean
Default: no

When set, mutt will skip the host name part of "6.3.74 ($hostname)" variable when adding the domain part to addresses. This variable does not affect the generation of Message-IDs, and it will not lead to the cut-off of first-level domains.

#### 6.3.67 hide_limited

Type: boolean
Default: no

When set, mutt will not show the presence of messages that are hidden by limiting, in the thread tree.

#### 6.3.68 hide_missing

Type: boolean
Default: yes

When set, mutt will not show the presence of missing messages in the thread tree.

#### 6.3.69 hide_thread_subject

Type: boolean
Default: yes

When set, mutt will not show the subject of messages in the thread tree that have the same subject as their parent or closest previously displayed sibling.

#### 6.3.70 hide_top_limited

Type: boolean
Default: no

When set, mutt will not show the presence of messages that are hidden by limiting, at the top of threads in the thread tree. Note that when 6.3.68 ($hide_missing) is set, this option will have no effect.

### 6.3.71 hide_top_missing

Type: boolean
Default: yes

When set, mutt will not show the presence of missing messages at the top of threads in the thread tree. Note that when 6.3.67 ($hide_limited) is set, this option will have no effect.

### 6.3.72 history

Type: number
Default: 10

This variable controls the size (in number of strings remembered) of the string history buffer. The buffer is cleared each time the variable is set.

### 6.3.73 honor_followup_to

Type: quadoption
Default: yes

This variable controls whether or not a Mail-Followup-To header is honored when group-replying to a message.

### 6.3.74 hostname

Type: string
Default: ""

Specifies the hostname to use after the "@" in local e-mail addresses. This overrides the compile time definition obtained from /etc/resolv.conf.

### 6.3.75 ignore_list_reply_to

Type: boolean
Default: no

Affects the behaviour of the *reply* function when replying to messages from mailing lists. When set, if the "Reply-To:" field is set to the same value as the "To:" field, Mutt assumes that the "Reply-To:" field was set by the mailing list to automate responses to the list, and will ignore this field. To direct a response to the mailing list when this option is set, use the *list-reply* function; *group-reply* will reply to both the sender and the list.

### 6.3.76 imap_authenticators

Type: string
Default: ""

This is a colon-delimited list of authentication methods mutt may attempt to use to log in to an IMAP server, in the order mutt should try them. Authentication methods are either 'login' or the right side of an IMAP

'AUTH=xxx' capability string, eg 'digest-md5', 'gssapi' or 'cram-md5'. This parameter is case-insensitive. If this parameter is unset (the default) mutt will try all available methods, in order from most-secure to least-secure.

Example: set imap_authenticators="gssapi:cram-md5:login"

**Note:** Mutt will only fall back to other authentication methods if the previous methods are unavailable. If a method is available but authentication fails, mutt will not connect to the IMAP server.

### 6.3.77 imap_delim_chars

Type: string
Default: "/."

This contains the list of characters which you would like to treat as folder separators for displaying IMAP paths. In particular it helps in using the '=' shortcut for your *folder* variable.

### 6.3.78 imap_force_ssl

Type: boolean
Default: no

If this variable is set, Mutt will always use SSL when connecting to IMAP servers.

### 6.3.79 imap_headers

Type: string
Default: ""

Mutt requests these header fields in addition to the default headers ("DATE FROM SUBJECT TO CC MESSAGE-ID REFERENCES CONTENT-TYPE CONTENT-DESCRIPTION IN-REPLY-TO REPLY-TO LINES X-LABEL") from IMAP servers before displaying the index menu. You may want to add more headers for spam detection. **Note:** This is a space separated list.

### 6.3.80 imap_home_namespace

Type: string
Default: ""

You normally want to see your personal folders alongside your INBOX in the IMAP browser. If you see something else, you may set this variable to the IMAP path to your folders.

### 6.3.81 imap_keepalive

Type: number
Default: 900

This variable specifies the maximum amount of time in seconds that mutt will wait before polling open IMAP connections, to prevent the server from closing them before mutt has finished with them. The default is well within the RFC-specified minimum amount of time (30 minutes) before a server is allowed to do this, but

in practice the RFC does get violated every now and then. Reduce this number if you find yourself getting disconnected from your IMAP server due to inactivity.

### 6.3.82 imap_list_subscribed

Type: boolean
Default: no

This variable configures whether IMAP folder browsing will look for only subscribed folders or all folders. This can be toggled in the IMAP browser with the *toggle-subscribed* function.

### 6.3.83 imap_pass

Type: string
Default: ""

Specifies the password for your IMAP account. If unset, Mutt will prompt you for your password when you invoke the fetch-mail function. **Warning**: you should only use this option when you are on a fairly secure machine, because the superuser can read your muttrc even if you are the only one who can read the file.

### 6.3.84 imap_passive

Type: boolean
Default: yes

When set, mutt will not open new IMAP connections to check for new mail. Mutt will only check for new mail over existing IMAP connections. This is useful if you don't want to be prompted to user/password pairs on mutt invocation, or if opening the connection is slow.

### 6.3.85 imap_peek

Type: boolean
Default: yes

If set, mutt will avoid implicitly marking your mail as read whenever you fetch a message from the server. This is generally a good thing, but can make closing an IMAP folder somewhat slower. This option exists to appease speed freaks.

### 6.3.86 imap_servernoise

Type: boolean
Default: yes

When set, mutt will display warning messages from the IMAP server as error messages. Since these messages are often harmless, or generated due to configuration problems on the server which are out of the users' hands, you may wish to suppress them at some point.

### 6.3.87 imap_user

Type: string
Default: ""

Your login name on the IMAP server.

This variable defaults to your user name on the local machine.

### 6.3.88 implicit_autoview

Type: boolean
Default: no

If set to "yes", mutt will look for a mailcap entry with the copiousoutput flag set for *every* MIME attachment it doesn't have an internal viewer defined for. If such an entry is found, mutt will use the viewer defined in that entry to convert the body part to text form.

### 6.3.89 include

Type: quadoption
Default: ask-yes

Controls whether or not a copy of the message(s) you are replying to is included in your reply.

### 6.3.90 include_onlyfirst

Type: boolean
Default: no

Controls whether or not Mutt includes only the first attachment of the message you are replying.

### 6.3.91 indent_string

Type: string
Default: "> "

Specifies the string to prepend to each line of text quoted in a message to which you are replying. You are strongly encouraged not to change this value, as it tends to agitate the more fanatical netizens.

### 6.3.92 index_format

Type: string
Default: "%4C %Z %{%b %d} %-15.15L (%?l?%4l&%4c?) %s"

This variable allows you to customize the message index display to your personal taste.

"Format strings" are similar to the strings used in the "C" function printf to format output (see the man page for more detail). The following sequences are defined in Mutt:

**%a**

address of the author

**%A**

reply-to address (if present; otherwise: address of author)

**%b**

filename of the original message folder (think mailBox)

**%B**

the list to which the letter was sent, or else the folder name (%b).

**%c**

number of characters (bytes) in the message

**%C**

current message number

**%d**

date and time of the message in the format specified by "date_format" converted to sender's time zone

**%D**

date and time of the message in the format specified by "date_format" converted to the local time zone

**%e**

current message number in thread

**%E**

number of messages in current thread

**%f**

entire From: line (address + real name)

**%F**

author name, or recipient name if the message is from you

**%H**

spam attribute(s) of this message

**%i**

message-id of the current message

**%l**

number of lines in the message (does not work with maildir, mh, and possibly IMAP folders)

**%L**

If an address in the To or CC header field matches an address defined by the users "subscribe" command, this displays "To <list-name>", otherwise the same as %F.

**%m**

total number of message in the mailbox

**%M**

number of hidden messages if the thread is collapsed.

**%N**

message score

**%n**

author's real name (or address if missing)

**%O**

(_O_riginal save folder) Where mutt would formerly have stashed the message: list name or recipient name if no list

**%s**

subject of the message

**%S**

status of the message (N/D/d/!/r/*)

**%t**

'to:' field (recipients)

**%T**

the appropriate character from the 6.3.265 ($to_chars) string

**%u**

user (login) name of the author

**%v**

first name of the author, or the recipient if the message is from you

**%y**

'x-label:' field, if present

**%Y**

'x-label' field, if present, and (1) not at part of a thread tree, (2) at the top of a thread, or (3) 'x-label' is different from preceding message's 'x-label'.

**%Z**

message status flags

**%{fmt}**

the date and time of the message is converted to sender's time zone, and "fmt" is expanded by the library function "strftime"; a leading bang disables locales

**%[fmt]**

the date and time of the message is converted to the local time zone, and "fmt" is expanded by the library function "strftime"; a leading bang disables locales

**%(fmt)**

the local date and time when the message was received. "fmt" is expanded by the library function "strftime"; a leading bang disables locales

**%<fmt>**

the current local time. "fmt" is expanded by the library function "strftime"; a leading bang disables locales.

**%>X**

right justify the rest of the string and pad with character "X"

**%|X**

pad to the end of the line with character "X"

See also: "6.3.265 ($to_chars)".

#### 6.3.93 ispell

Type: path
Default: "/usr/bin/ispell"

How to invoke ispell (GNU's spell-checking software).

#### 6.3.94 keep_flagged

Type: boolean
Default: no

If set, read messages marked as flagged will not be moved from your spool mailbox to your "6.3.103 ($mbox)" mailbox, or as a result of a "3.11 (mbox-hook)" command.

#### 6.3.95 locale

Type: string
Default: "C"

The locale used by *strftime(3)* to format dates. Legal values are the strings your system accepts for the locale variable *LC_TIME*.

#### 6.3.96 mail_check

Type: number
Default: 5

This variable configures how often (in seconds) mutt should look for new mail.

### 6.3.97 mailcap_path

Type: string
Default: ""

This variable specifies which files to consult when attempting to display MIME bodies not directly supported by Mutt.

### 6.3.98 mailcap_sanitize

Type: boolean
Default: yes

If set, mutt will restrict possible characters in mailcap % expandos to a well-defined set of safe characters. This is the safe setting, but we are not sure it doesn't break some more advanced MIME stuff.

**DON'T CHANGE THIS SETTING UNLESS YOU ARE REALLY SURE WHAT YOU ARE DOING!**

### 6.3.99 maildir_trash

Type: boolean
Default: no

If set, messages marked as deleted will be saved with the maildir (T)rashed flag instead of unlinked. **NOTE:** this only applies to maildir-style mailboxes. Setting it will have no effect on other mailbox types.

### 6.3.100 mark_old

Type: boolean
Default: yes

Controls whether or not mutt marks *new* **unread** messages as *old* if you exit a mailbox without reading them. With this option set, the next time you start mutt, the messages will show up with an "O" next to them in the index menu, indicating that they are old.

### 6.3.101 markers

Type: boolean
Default: yes

Controls the display of wrapped lines in the internal pager. If set, a "+" marker is displayed at the beginning of wrapped lines. Also see the "6.3.244 ($smart_wrap)" variable.

### 6.3.102 mask

Type: regular expression
Default: "!^\.[^.]"

A regular expression used in the file browser, optionally preceded by the *not* operator "!". Only files whose names match this mask will be shown. The match is always case-sensitive.

#### 6.3.103 mbox

Type: path
Default: "~/mbox"

This specifies the folder into which read mail in your "6.3.253 ($spoolfile)" folder will be appended.

#### 6.3.104 mbox_type

Type: folder magic
Default: mbox

The default mailbox type used when creating new folders. May be any of mbox, MMDF, MH and Maildir.

#### 6.3.105 metoo

Type: boolean
Default: no

If unset, Mutt will remove your address (see the "alternates" command) from the list of recipients when replying to a message.

#### 6.3.106 menu_context

Type: number
Default: 0

This variable controls the number of lines of context that are given when scrolling through menus. (Similar to "6.3.123 ($pager_context)".)

#### 6.3.107 menu_move_off

Type: boolean
Default: no

When *unset*, the bottom entry of menus will never scroll up past the bottom of the screen, unless there are less entries than lines. When *set*, the bottom entry may move off the bottom.

#### 6.3.108 menu_scroll

Type: boolean
Default: no

When *set*, menus will be scrolled up or down one line when you attempt to move across a screen boundary. If *unset*, the screen is cleared and the next or previous page of the menu is displayed (useful for slow links to avoid many redraws).

### 6.3.109 meta_key

Type: boolean
Default: no

If set, forces Mutt to interpret keystrokes with the high bit (bit 8) set as if the user had pressed the ESC key and whatever key remains after having the high bit removed. For example, if the key pressed has an ASCII value of 0xf4, then this is treated as if the user had pressed ESC then "x". This is because the result of removing the high bit from "0xf4" is "0x74", which is the ASCII character "x".

### 6.3.110 mh_purge

Type: boolean
Default: no

When unset, mutt will mimic mh's behaviour and rename deleted messages to *,<old file name>* in mh folders instead of really deleting them. If the variable is set, the message files will simply be deleted.

### 6.3.111 mh_seq_flagged

Type: string
Default: "flagged"

The name of the MH sequence used for flagged messages.

### 6.3.112 mh_seq_replied

Type: string
Default: "replied"

The name of the MH sequence used to tag replied messages.

### 6.3.113 mh_seq_unseen

Type: string
Default: "unseen"

The name of the MH sequence used for unseen messages.

### 6.3.114 mime_forward

Type: quadoption
Default: no

When set, the message you are forwarding will be attached as a separate MIME part instead of included in the main body of the message. This is useful for forwarding MIME messages so the receiver can properly view the message as it was delivered to you. If you like to switch between MIME and not MIME from mail to mail, set this variable to ask-no or ask-yes.

Also see "6.3.57 ($forward_decode)" and "6.3.115 ($mime_forward_decode)".

### 6.3.115 mime_forward_decode

Type: boolean
Default: no

Controls the decoding of complex MIME messages into text/plain when forwarding a message while "6.3.114 ($mime_forward)" is *set*. Otherwise "6.3.57 ($forward_decode)" is used instead.

### 6.3.116 mime_forward_rest

Type: quadoption
Default: yes

When forwarding multiple attachments of a MIME message from the recvattach menu, attachments which cannot be decoded in a reasonable manner will be attached to the newly composed message if this option is set.

### 6.3.117 mix_entry_format

Type: string
Default: "%4n %c %-16s %a"

This variable describes the format of a remailer line on the mixmaster chain selection screen. The following printf-like sequences are supported:

**%n**

The running number on the menu.

**%c**

Remailer capabilities.

**%s**

The remailer's short name.

**%a**

The remailer's e-mail address.

### 6.3.118 mixmaster

Type: path
Default: "mixmaster"

This variable contains the path to the Mixmaster binary on your system. It is used with various sets of parameters to gather the list of known remailers, and to finally send a message through the mixmaster chain.

#### 6.3.119 move

Type: quadoption
Default: ask-no

Controls whether or not Mutt will move read messages from your spool mailbox to your "6.3.103 ($mbox)" mailbox, or as a result of a "3.11 (mbox-hook)" command.

#### 6.3.120 message_format

Type: string
Default: "%s"

This is the string displayed in the "attachment" menu for attachments of type message/rfc822. For a full listing of defined printf()-like sequences see the section on "6.3.92 ($index_format)".

#### 6.3.121 narrow_tree

Type: boolean
Default: no

This variable, when set, makes the thread tree narrower, allowing deeper threads to fit on the screen.

#### 6.3.122 pager

Type: path
Default: "builtin"

This variable specifies which pager you would like to use to view messages. builtin means to use the builtin pager, otherwise this variable should specify the pathname of the external pager you would like to use.

Using an external pager may have some disadvantages: Additional keystrokes are necessary because you can't call mutt functions directly from the pager, and screen resizes cause lines longer than the screen width to be badly formatted in the help menu.

#### 6.3.123 pager_context

Type: number
Default: 0

This variable controls the number of lines of context that are given when displaying the next or previous page in the internal pager. By default, Mutt will display the line after the last one on the screen at the top of the next page (0 lines of context).

#### 6.3.124 pager_format

Type: string
Default: "-%Z- %C/%m: %-20.20n %s"

This variable controls the format of the one-line message "status" displayed before each message in either the internal or an external pager. The valid sequences are listed in the "6.3.92 ($index_format)" section.

### 6.3.125 pager_index_lines

Type: number
Default: 0

Determines the number of lines of a mini-index which is shown when in the pager. The current message, unless near the top or bottom of the folder, will be roughly one third of the way down this mini-index, giving the reader the context of a few messages before and after the message. This is useful, for example, to determine how many messages remain to be read in the current thread. One of the lines is reserved for the status bar from the index, so a *pager_index_lines* of 6 will only show 5 lines of the actual index. A value of 0 results in no index being shown. If the number of messages in the current folder is less than *pager_index_lines*, then the index will only use as many lines as it needs.

### 6.3.126 pager_stop

Type: boolean
Default: no

When set, the internal-pager will **not** move to the next message when you are at the end of a message and invoke the *next-page* function.

### 6.3.127 crypt_autosign

Type: boolean
Default: no

Setting this variable will cause Mutt to always attempt to cryptographically sign outgoing messages. This can be overridden by use of the *pgp-menu*, when signing is not required or encryption is requested as well. If "6.3.136 ($smime_is_default)" is set, then OpenSSL is used instead to create S/MIME messages and settings can be overridden by use of the *smime-menu*. (Crypto only)

### 6.3.128 crypt_autoencrypt

Type: boolean
Default: no

Setting this variable will cause Mutt to always attempt to PGP encrypt outgoing messages. This is probably only useful in connection to the *send-hook* command. It can be overridden by use of the *pgp-menu*, when encryption is not required or signing is requested as well. IF "6.3.136 ($smime_is_default)" is set, then OpenSSL is used instead to create S/MIME messages and settings can be overridden by use of the *smime-menu*. (Crypto only)

#### 6.3.129 pgp_ignore_subkeys

Type: boolean
Default: yes

Setting this variable will cause Mutt to ignore OpenPGP subkeys. Instead, the principal key will inherit the subkeys' capabilities. Unset this if you want to play interesting key selection games. (PGP only)

#### 6.3.130 crypt_replyencrypt

Type: boolean
Default: yes

If set, automatically PGP or OpenSSL encrypt replies to messages which are encrypted. (Crypto only)

#### 6.3.131 crypt_replysign

Type: boolean
Default: no

If set, automatically PGP or OpenSSL sign replies to messages which are signed.

**Note:** this does not work on messages that are encrypted **and** signed! (Crypto only)

#### 6.3.132 crypt_replysignencrypted

Type: boolean
Default: no

If set, automatically PGP or OpenSSL sign replies to messages which are encrypted. This makes sense in combination with "6.3.130 ($crypt_replyencrypt)", because it allows you to sign all messages which are automatically encrypted. This works around the problem noted in "6.3.131 ($crypt_replysign)", that mutt is not able to find out whether an encrypted message is also signed. (Crypto only)

#### 6.3.133 crypt_timestamp

Type: boolean
Default: yes

If set, mutt will include a time stamp in the lines surrounding PGP or S/MIME output, so spoofing such lines is more difficult. If you are using colors to mark these lines, and rely on these, you may unset this setting. (Crypto only)

#### 6.3.134 pgp_use_gpg_agent

Type: boolean
Default: no

If set, mutt will use a possibly-running gpg-agent process. (PGP only)

#### 6.3.135 crypt_verify_sig

Type: quadoption
Default: yes

If "yes", always attempt to verify PGP or S/MIME signatures. If "ask", ask whether or not to verify the signature. If "no", never attempt to verify cryptographic signatures. (Crypto only)

#### 6.3.136 smime_is_default

Type: boolean
Default: no

The default behaviour of mutt is to use PGP on all auto-sign/encryption operations. To override and to use OpenSSL instead this must be set. However, this has no effect while replying, since mutt will automatically select the same application that was used to sign/encrypt the original message. (Note that this variable can be overridden by unsetting 6.3.34 ($crypt_autosmime).) (S/MIME only)

#### 6.3.137 smime_ask_cert_label

Type: boolean
Default: yes

This flag controls whether you want to be asked to enter a label for a certificate about to be added to the database or not. It is set by default. (S/MIME only)

#### 6.3.138 smime_decrypt_use_default_key

Type: boolean
Default: yes

If set (default) this tells mutt to use the default key for decryption. Otherwise, if manage multiple certificate-key-pairs, mutt will try to use the mailbox-address to determine the key to use. It will ask you to supply a key, if it can't find one. (S/MIME only)

#### 6.3.139 pgp_entry_format

Type: string
Default: "%4n %t%f %4l/0x%k %-4a %2c %u"

This variable allows you to customize the PGP key selection menu to your personal taste. This string is similar to "6.3.92 ($index_format)", but has its own set of printf()-like sequences:

**%n**

number

**%k**

key id

**%u**

user id

**%a**

algorithm

**%l**

key length

**%f**

flags

**%c**

capabilities

**%t**

trust/validity of the key-uid association

**%[<s>]**

date of the key where <s> is an strftime(3) expression

(PGP only)

#### 6.3.140 pgp_good_sign

Type: regular expression
Default: ""

If you assign a text to this variable, then a PGP signature is only considered verified if the output from 6.3.155 ($pgp_verify_command) contains the text. Use this variable if the exit code from the command is 0 even for bad signatures. (PGP only)

#### 6.3.141 pgp_check_exit

Type: boolean
Default: yes

If set, mutt will check the exit code of the PGP subprocess when signing or encrypting. A non-zero exit code means that the subprocess failed. (PGP only)

#### 6.3.142 pgp_long_ids

Type: boolean
Default: no

If set, use 64 bit PGP key IDs. Unset uses the normal 32 bit Key IDs. (PGP only)

#### 6.3.143 pgp_retainable_sigs

Type: boolean
Default: no

If set, signed and encrypted messages will consist of nested multipart/signed and multipart/encrypted body parts.

This is useful for applications like encrypted and signed mailing lists, where the outer layer (multipart/encrypted) can be easily removed, while the inner multipart/signed part is retained. (PGP only)

#### 6.3.144 pgp_autoinline

Type: boolean
Default: no

This option controls whether Mutt generates old-style inline (traditional) PGP encrypted or signed messages under certain circumstances. This can be overridden by use of the *pgp-menu*, when inline is not required.

Note that Mutt might automatically use PGP/MIME for messages which consist of more than a single MIME part. Mutt can be configured to ask before sending PGP/MIME messages when inline (traditional) would not work. See also: "6.3.151 ($pgp_mime_auto)".

Also note that using the old-style PGP message format is **strongly deprecated**. (PGP only)

#### 6.3.145 pgp_replyinline

Type: boolean
Default: no

Setting this variable will cause Mutt to always attempt to create an inline (traditional) message when replying to a message which is PGP encrypted/signed inline. This can be overridden by use of the *pgp-menu*, when inline is not required. This option does not automatically detect if the (replied-to) message is inline; instead it relies on Mutt internals for previously checked/flagged messages.

Note that Mutt might automatically use PGP/MIME for messages which consist of more than a single MIME part. Mutt can be configured to ask before sending PGP/MIME messages when inline (traditional) would not work. See also: "6.3.151 ($pgp_mime_auto)".

Also note that using the old-style PGP message format is **strongly deprecated**. (PGP only)

#### 6.3.146 pgp_show_unusable

Type: boolean
Default: yes

If set, mutt will display non-usable keys on the PGP key selection menu. This includes keys which have been revoked, have expired, or have been marked as "disabled" by the user. (PGP only)

#### 6.3.147 pgp_sign_as

Type: string
Default: ""

If you have more than one key pair, this option allows you to specify which of your private keys to use. It is recommended that you use the keyid form to specify your key (e.g., "0x00112233"). (PGP only)

#### 6.3.148 pgp_strict_enc

Type: boolean
Default: yes

If set, Mutt will automatically encode PGP/MIME signed messages as *quoted-printable*. Please note that unsetting this variable may lead to problems with non-verifyable PGP signatures, so only change this if you know what you are doing. (PGP only)

#### 6.3.149 pgp_timeout

Type: number
Default: 300

The number of seconds after which a cached passphrase will expire if not used. (PGP only)

#### 6.3.150 pgp_sort_keys

Type: sort order
Default: address

Specifies how the entries in the 'pgp keys' menu are sorted. The following are legal values:

**address**

sort alphabetically by user id

**keyid**

sort alphabetically by key id

**date**

sort by key creation date

**trust**

sort by the trust of the key

If you prefer reverse order of the above values, prefix it with 'reverse-'. (PGP only)

### 6.3.151 pgp_mime_auto

Type: quadoption
Default: ask-yes

This option controls whether Mutt will prompt you for automatically sending a (signed/encrypted) message using PGP/MIME when inline (traditional) fails (for any reason).

Also note that using the old-style PGP message format is **strongly deprecated**. (PGP only)

### 6.3.152 pgp_auto_decode

Type: boolean
Default: no

If set, mutt will automatically attempt to decrypt traditional PGP messages whenever the user performs an operation which ordinarily would result in the contents of the message being operated on. For example, if the user displays a pgp-traditional message which has not been manually checked with the check-traditional-pgp function, mutt will automatically check the message for traditional pgp.

### 6.3.153 pgp_decode_command

Type: string
Default: ""

This format strings specifies a command which is used to decode application/pgp attachments.

The PGP command formats have their own set of printf-like sequences:

**%p**

Expands to PGPPASSFD=0 when a pass phrase is needed, to an empty string otherwise. Note: This may be used with a %? construct.

**%f**

Expands to the name of a file containing a message.

**%s**

Expands to the name of a file containing the signature part of a multipart/signed attachment when verifying it.

**%a**

The value of 6.3.147 ($pgp_sign_as).

**%r**

One or more key IDs.

For examples on how to configure these formats for the various versions of PGP which are floating around, see the pgp*.rc and gpg.rc files in the samples/ subdirectory which has been installed on your system alongside the documentation. (PGP only)

#### 6.3.154 pgp_getkeys_command

Type: string
Default: ""

This command is invoked whenever mutt will need public key information. %r is the only printf-like sequence used with this format. (PGP only)

#### 6.3.155 pgp_verify_command

Type: string
Default: ""

This command is used to verify PGP signatures. (PGP only)

#### 6.3.156 pgp_decrypt_command

Type: string
Default: ""

This command is used to decrypt a PGP encrypted message. (PGP only)

#### 6.3.157 pgp_clearsign_command

Type: string
Default: ""

This format is used to create a old-style "clearsigned" PGP message. Note that the use of this format is **strongly deprecated**. (PGP only)

#### 6.3.158 pgp_sign_command

Type: string
Default: ""

This command is used to create the detached PGP signature for a multipart/signed PGP/MIME body part. (PGP only)

#### 6.3.159 pgp_encrypt_sign_command

Type: string
Default: ""

This command is used to both sign and encrypt a body part. (PGP only)

#### 6.3.160 pgp_encrypt_only_command

Type: string
Default: ""

This command is used to encrypt a body part without signing it. (PGP only)

#### 6.3.161 pgp_import_command

Type: string
Default: ""

This command is used to import a key from a message into the user's public key ring. (PGP only)

#### 6.3.162 pgp_export_command

Type: string
Default: ""

This command is used to export a public key from the user's key ring. (PGP only)

#### 6.3.163 pgp_verify_key_command

Type: string
Default: ""

This command is used to verify key information from the key selection menu. (PGP only)

#### 6.3.164 pgp_list_secring_command

Type: string
Default: ""

This command is used to list the secret key ring's contents. The output format must be analogous to the one used by gpg –list-keys –with-colons.

This format is also generated by the pgpring utility which comes with mutt. (PGP only)

#### 6.3.165 pgp_list_pubring_command

Type: string
Default: ""

This command is used to list the public key ring's contents. The output format must be analogous to the one used by gpg –list-keys –with-colons.

This format is also generated by the pgpring utility which comes with mutt. (PGP only)

#### 6.3.166 forward_decrypt

Type: boolean
Default: yes

Controls the handling of encrypted messages when forwarding a message. When set, the outer layer of encryption is stripped off. This variable is only used if "6.3.114 ($mime_forward)" is *set* and "6.3.115 ($mime_forward_decode)" is *unset*. (PGP only)

#### 6.3.167 smime_timeout

Type: number
Default: 300

The number of seconds after which a cached passphrase will expire if not used. (S/MIME only)

#### 6.3.168 smime_encrypt_with

Type: string
Default: ""

This sets the algorithm that should be used for encryption. Valid choices are "des", "des3", "rc2-40", "rc2-64", "rc2-128". If unset "3des" (TripleDES) is used. (S/MIME only)

#### 6.3.169 smime_keys

Type: path
Default: ""

Since there is no pubring/secring as with PGP, mutt has to handle storage ad retrieval of keys/certs by itself. This is very basic right now, and stores keys and certificates in two different directories, both named as the hash-value retrieved from OpenSSL. There is an index file which contains mailbox-address keyid pair, and which can be manually edited. This one points to the location of the private keys. (S/MIME only)

#### 6.3.170 smime_ca_location

Type: path
Default: ""

This variable contains the name of either a directory, or a file which contains trusted certificates for use with OpenSSL. (S/MIME only)

#### 6.3.171 smime_certificates

Type: path
Default: ""

Since there is no pubring/secring as with PGP, mutt has to handle storage and retrieval of keys by itself. This is very basic right now, and keys and certificates are stored in two different directories, both named as the hash-value retrieved from OpenSSL. There is an index file which contains mailbox-address keyid pairs, and which can be manually edited. This one points to the location of the certificates. (S/MIME only)

#### 6.3.172 smime_decrypt_command

Type: string
Default: ""

This format string specifies a command which is used to decrypt application/x-pkcs7-mime attachments.

The OpenSSL command formats have their own set of printf-like sequences similar to PGP's:

**%f**

Expands to the name of a file containing a message.

**%s**

Expands to the name of a file containing the signature part of a multipart/signed attachment when verifying it.

**%k**

The key-pair specified with 6.3.183 ($smime_default_key)

**%c**

One or more certificate IDs.

**%a**

The algorithm used for encryption.

**%C**

CA location: Depending on whether 6.3.170 ($smime_ca_location) points to a directory or file, this expands to "-CApath 6.3.170 ($smime_ca_location)" or "-CAfile 6.3.170 ($smime_ca_location)".

For examples on how to configure these formats, see the smime.rc in the samples/ subdirectory which has been installed on your system alongside the documentation. (S/MIME only)

### 6.3.173 smime_verify_command

Type: string
Default: ""

This command is used to verify S/MIME signatures of type multipart/signed. (S/MIME only)

### 6.3.174 smime_verify_opaque_command

Type: string
Default: ""

This command is used to verify S/MIME signatures of type application/x-pkcs7-mime. (S/MIME only)

### 6.3.175 smime_sign_command

Type: string
Default: ""

This command is used to created S/MIME signatures of type multipart/signed, which can be read by all mail clients. (S/MIME only)

#### 6.3.176 smime_sign_opaque_command

Type: string
Default: ""

This command is used to created S/MIME signatures of type application/x-pkcs7-signature, which can only be handled by mail clients supporting the S/MIME extension. (S/MIME only)

#### 6.3.177 smime_encrypt_command

Type: string
Default: ""

This command is used to create encrypted S/MIME messages. (S/MIME only)

#### 6.3.178 smime_pk7out_command

Type: string
Default: ""

This command is used to extract PKCS7 structures of S/MIME signatures, in order to extract the public X509 certificate(s). (S/MIME only)

#### 6.3.179 smime_get_cert_command

Type: string
Default: ""

This command is used to extract X509 certificates from a PKCS7 structure. (S/MIME only)

#### 6.3.180 smime_get_signer_cert_command

Type: string
Default: ""

This command is used to extract only the signers X509 certificate from a S/MIME signature, so that the certificate's owner may get compared to the email's 'From'-field. (S/MIME only)

#### 6.3.181 smime_import_cert_command

Type: string
Default: ""

This command is used to import a certificate via smime_keys. (S/MIME only)

#### 6.3.182 smime_get_cert_email_command

Type: string
Default: ""

This command is used to extract the mail address(es) used for storing X509 certificates, and for verification purposes (to check whether the certificate was issued for the sender's mailbox). (S/MIME only)

#### 6.3.183 smime_default_key

Type: string
Default: ""

This is the default key-pair to use for signing. This must be set to the keyid (the hash-value that OpenSSL generates) to work properly (S/MIME only)

#### 6.3.184 ssl_client_cert

Type: path
Default: ""

The file containing a client certificate and its associated private key.

#### 6.3.185 ssl_starttls

Type: quadoption
Default: yes

If set (the default), mutt will attempt to use STARTTLS on servers advertising the capability. When unset, mutt will not attempt to use STARTTLS regardless of the server's capabilities.

#### 6.3.186 certificate_file

Type: path
Default: "~/.mutt_certificates"

This variable specifies the file where the certificates you trust are saved. When an unknown certificate is encountered, you are asked if you accept it or not. If you accept it, the certificate can also be saved in this file and further connections are automatically accepted.

You can also manually add CA certificates in this file. Any server certificate that is signed with one of these CA certificates are also automatically accepted.

Example: set certificate_file=~/.mutt/certificates

#### 6.3.187 ssl_usesystemcerts

Type: boolean
Default: yes

If set to *yes*, mutt will use CA certificates in the system-wide certificate store when checking if server certificate is signed by a trusted CA.

#### 6.3.188 entropy_file

Type: path
Default: ""

The file which includes random data that is used to initialize SSL library functions.

#### 6.3.189 ssl_use_sslv2

Type: boolean
Default: yes

This variables specifies whether to attempt to use SSLv2 in the SSL authentication process.

#### 6.3.190 ssl_use_sslv3

Type: boolean
Default: yes

This variables specifies whether to attempt to use SSLv3 in the SSL authentication process.

#### 6.3.191 ssl_use_tlsv1

Type: boolean
Default: yes

This variables specifies whether to attempt to use TLSv1 in the SSL authentication process.

#### 6.3.192 pipe_split

Type: boolean
Default: no

Used in connection with the *pipe-message* command and the "tag- prefix" operator. If this variable is unset, when piping a list of tagged messages Mutt will concatenate the messages and will pipe them as a single folder. When set, Mutt will pipe the messages one by one. In both cases the messages are piped in the current sorted order, and the "6.3.194 ($pipe_sep)" separator is added after each message.

#### 6.3.193 pipe_decode

Type: boolean
Default: no

Used in connection with the *pipe-message* command. When unset, Mutt will pipe the messages without any preprocessing. When set, Mutt will weed headers and will attempt to PGP/MIME decode the messages first.

#### 6.3.194 pipe_sep

Type: string
Default: "\n"

The separator to add between messages when piping a list of tagged messages to an external Unix command.

#### 6.3.195 pop_authenticators

Type: string
Default: ""

This is a colon-delimited list of authentication methods mutt may attempt to use to log in to an POP server, in the order mutt should try them. Authentication methods are either 'user', 'apop' or any SASL mechanism, eg 'digest-md5', 'gssapi' or 'cram-md5'. This parameter is case-insensitive. If this parameter is unset (the default) mutt will try all available methods, in order from most-secure to least-secure.

Example: set pop_authenticators="digest-md5:apop:user"

#### 6.3.196 pop_auth_try_all

Type: boolean
Default: yes

If set, Mutt will try all available methods. When unset, Mutt will only fall back to other authentication methods if the previous methods are unavailable. If a method is available but authentication fails, Mutt will not connect to the POP server.

#### 6.3.197 pop_checkinterval

Type: number
Default: 60

This variable configures how often (in seconds) POP should look for new mail.

#### 6.3.198 pop_delete

Type: quadoption
Default: ask-no

If set, Mutt will delete successfully downloaded messages from the POP server when using the fetch-mail function. When unset, Mutt will download messages but also leave them on the POP server.

#### 6.3.199 pop_host

Type: string
Default: ""

The name of your POP server for the fetch-mail function. You can also specify an alternative port, username and password, ie:

[pop[s]://][username[:password]@]popserver[:port]

#### 6.3.200 pop_last

Type: boolean
Default: no

If this variable is set, mutt will try to use the "LAST" POP command for retrieving only unread messages from the POP server when using the fetch-mail function.

#### 6.3.201 pop_reconnect

Type: quadoption
Default: ask-yes

Controls whether or not Mutt will try to reconnect to POP server when connection lost.

#### 6.3.202 pop_user

Type: string
Default: ""

Your login name on the POP server.

This variable defaults to your user name on the local machine.

#### 6.3.203 pop_pass

Type: string
Default: ""

Specifies the password for your POP account. If unset, Mutt will prompt you for your password when you open POP mailbox. **Warning**: you should only use this option when you are on a fairly secure machine, because the superuser can read your muttrc even if you are the only one who can read the file.

#### 6.3.204 post_indent_string

Type: string
Default: ""

Similar to the "6.3.14 ($attribution)" variable, Mutt will append this string after the inclusion of a message which is being replied to.

#### 6.3.205 postpone

Type: quadoption
Default: ask-yes

Controls whether or not messages are saved in the "6.3.206 ($postponed)" mailbox when you elect not to send immediately.

### 6.3.206 postponed

Type: path
Default: "~/postponed"

Mutt allows you to indefinitely "6.3.205 (postpone) sending a message" which you are editing. When you choose to postpone a message, Mutt saves it in the mailbox specified by this variable. Also see the "6.3.205 ($postpone)" variable.

### 6.3.207 preconnect

Type: string
Default: ""

If set, a shell command to be executed if mutt fails to establish a connection to the server. This is useful for setting up secure connections, e.g. with ssh(1). If the command returns a nonzero status, mutt gives up opening the server. Example:

preconnect="ssh -f -q -L 1234:mailhost.net:143 mailhost.net sleep 20 < /dev/null > /dev/null"

Mailbox 'foo' on mailhost.net can now be reached as '{localhost:1234}foo'.

NOTE: For this example to work, you must be able to log in to the remote machine without having to enter a password.

### 6.3.208 print

Type: quadoption
Default: ask-no

Controls whether or not Mutt really prints messages. This is set to *ask-no* by default, because some people accidentally hit "p" often (like me).

### 6.3.209 print_command

Type: path
Default: "lpr"

This specifies the command pipe that should be used to print messages.

### 6.3.210 print_decode

Type: boolean
Default: yes

Used in connection with the print-message command. If this option is set, the message is decoded before it is passed to the external command specified by 6.3.209 ($print_command). If this option is unset, no processing will be applied to the message when printing it. The latter setting may be useful if you are using some advanced printer filter which is able to properly format e-mail messages for printing.

### 6.3.211 print_split

Type: boolean
Default: no

Used in connection with the print-message command. If this option is set, the command specified by 6.3.209 ($print_command) is executed once for each message which is to be printed. If this option is unset, the command specified by 6.3.209 ($print_command) is executed only once, and all the messages are concatenated, with a form feed as the message separator.

Those who use the **enscript**(1) program's mail-printing mode will most likely want to set this option.

### 6.3.212 prompt_after

Type: boolean
Default: yes

If you use an *external* "6.3.122 ($pager)", setting this variable will cause Mutt to prompt you for a command when the pager exits rather than returning to the index menu. If unset, Mutt will return to the index menu when the external pager exits.

### 6.3.213 query_command

Type: path
Default: ""

This specifies the command that mutt will use to make external address queries. The string should contain a %s, which will be substituted with the query string the user types. See "4.5 (query)" for more information.

### 6.3.214 quit

Type: quadoption
Default: yes

This variable controls whether "quit" and "exit" actually quit from mutt. If it set to yes, they do quit, if it is set to no, they have no effect, and if it is set to ask-yes or ask-no, you are prompted for confirmation when you try to quit.

### 6.3.215 quote_regexp

Type: regular expression
Default: "^([ \t]*[|>:}#])+"

A regular expression used in the internal-pager to determine quoted sections of text in the body of a message.

**Note:** In order to use the *quoted***x** patterns in the internal pager, you need to set this to a regular expression that matches *exactly* the quote characters at the beginning of quoted lines.

#### 6.3.216 read_inc

Type: number
Default: 10

If set to a value greater than 0, Mutt will display which message it is currently on when reading a mailbox. The message is printed after *read_inc* messages have been read (e.g., if set to 25, Mutt will print a message when it reads message 25, and then again when it gets to message 50). This variable is meant to indicate progress when reading large mailboxes which may take some time. When set to 0, only a single message will appear before the reading the mailbox.

Also see the "6.3.278 ($write_inc)" variable.

#### 6.3.217 read_only

Type: boolean
Default: no

If set, all folders are opened in read-only mode.

#### 6.3.218 realname

Type: string
Default: ""

This variable specifies what "real" or "personal" name should be used when sending messages.

By default, this is the GECOS field from /etc/passwd. Note that this variable will *not* be used when the user has set a real name in the 6.3.61 ($from) variable.

#### 6.3.219 recall

Type: quadoption
Default: ask-yes

Controls whether or not Mutt recalls postponed messages when composing a new message. Also see "6.3.206 ($postponed)".

Setting this variable to "yes" is not generally useful, and thus not recommended.

#### 6.3.220 record

Type: path
Default: ""

This specifies the file into which your outgoing messages should be appended. (This is meant as the primary method for saving a copy of your messages, but another way to do this is using the "3.13 (my_hdr)" command to create a *Bcc:* field with your email address in it.)

The value of *6.3.220 ($record)* is overridden by the "6.3.56 ($force_name)" and "6.3.231 ($save_name)" variables, and the "3.16 (fcc-hook)" command.

#### 6.3.221 reply_regexp

Type: regular expression
Default: "^(re([\[0-9\]+])*|aw):[ \t]*"

A regular expression used to recognize reply messages when threading and replying. The default value corresponds to the English "Re:" and the German "Aw:".

#### 6.3.222 reply_self

Type: boolean
Default: no

If unset and you are replying to a message sent by you, Mutt will assume that you want to reply to the recipients of that message rather than to yourself.

#### 6.3.223 reply_to

Type: quadoption
Default: ask-yes

If set, when replying to a message, Mutt will use the address listed in the Reply-to: header as the recipient of the reply. If unset, it will use the address in the From: header field instead. This option is useful for reading a mailing list that sets the Reply-To: header field to the list address and you want to send a private message to the author of a message.

#### 6.3.224 resolve

Type: boolean
Default: yes

When set, the cursor will be automatically advanced to the next (possibly undeleted) message whenever a command that modifies the current message is executed.

#### 6.3.225 reverse_alias

Type: boolean
Default: no

This variable controls whether or not Mutt will display the "personal" name from your aliases in the index menu if it finds an alias that matches the message's sender. For example, if you have the following alias:

```
alias juser abd30425@somewhere.net (Joe User)
```

and then you receive mail which contains the following header:

```
From: abd30425@somewhere.net
```

It would be displayed in the index menu as "Joe User" instead of "abd30425@somewhere.net." This is useful when the person's e-mail address is not human friendly (like CompuServe addresses).

#### 6.3.226 reverse_name

Type: boolean
Default: no

It may sometimes arrive that you receive mail to a certain machine, move the messages to another machine, and reply to some the messages from there. If this variable is set, the default *From:* line of the reply messages is built using the address where you received the messages you are replying to **if** that address matches your alternates. If the variable is unset, or the address that would be used doesn't match your alternates, the *From:* line will use your address on the current machine.

#### 6.3.227 reverse_realname

Type: boolean
Default: yes

This variable fine-tunes the behaviour of the 6.3.226 (reverse_name) feature. When it is set, mutt will use the address from incoming messages as-is, possibly including eventual real names. When it is unset, mutt will override any such real names with the setting of the 6.3.218 (realname) variable.

#### 6.3.228 rfc2047_parameters

Type: boolean
Default: no

When this variable is set, Mutt will decode RFC-2047-encoded MIME parameters. You want to set this variable when mutt suggests you to save attachments to files named like this: =?iso-8859-1?Q?file=5F=E4=5F991116=2Ezip?=

When this variable is set interactively, the change doesn't have the desired effect before you have changed folders.

Note that this use of RFC 2047's encoding is explicitly, prohibited by the standard, but nevertheless encountered in the wild. Also note that setting this parameter will *not* have the effect that mutt *generates* this kind of encoding. Instead, mutt will unconditionally use the encoding specified in RFC 2231.

#### 6.3.229 save_address

Type: boolean
Default: no

If set, mutt will take the sender's full address when choosing a default folder for saving a mail. If "6.3.231 ($save_name)" or "6.3.56 ($force_name)" is set too, the selection of the fcc folder will be changed as well.

#### 6.3.230 save_empty

Type: boolean
Default: yes

When unset, mailboxes which contain no saved messages will be removed when closed (the exception is "6.3.253 ($spoolfile)" which is never removed). If set, mailboxes are never removed.

**Note:** This only applies to mbox and MMDF folders, Mutt does not delete MH and Maildir directories.

#### 6.3.231 save_name

Type: boolean
Default: no

This variable controls how copies of outgoing messages are saved. When set, a check is made to see if a mailbox specified by the recipient address exists (this is done by searching for a mailbox in the "6.3.53 ($folder)" directory with the *username* part of the recipient address). If the mailbox exists, the outgoing message will be saved to that mailbox, otherwise the message is saved to the "6.3.220 ($record)" mailbox.

Also see the "6.3.56 ($force_name)" variable.

#### 6.3.232 score

Type: boolean
Default: yes

When this variable is *unset*, scoring is turned off. This can be useful to selectively disable scoring for certain folders when the "6.3.233 ($score_threshold_delete)" variable and friends are used.

#### 6.3.233 score_threshold_delete

Type: number
Default: -1

Messages which have been assigned a score equal to or lower than the value of this variable are automatically marked for deletion by mutt. Since mutt scores are always greater than or equal to zero, the default setting of this variable will never mark a message for deletion.

#### 6.3.234 score_threshold_flag

Type: number
Default: 9999

Messages which have been assigned a score greater than or equal to this variable's value are automatically marked "flagged".

#### 6.3.235 score_threshold_read

Type: number
Default: -1

Messages which have been assigned a score equal to or lower than the value of this variable are automatically marked as read by mutt. Since mutt scores are always greater than or equal to zero, the default setting of this variable will never mark a message read.

### 6.3.236 send_charset

Type: string
Default: "us-ascii:iso-8859-1:utf-8"

A list of character sets for outgoing messages. Mutt will use the first character set into which the text can be converted exactly. If your "6.3.21 ($charset)" is not iso-8859-1 and recipients may not understand UTF-8, it is advisable to include in the list an appropriate widely used standard character set (such as iso-8859-2, koi8-r or iso-2022-jp) either instead of or after "iso-8859-1".

### 6.3.237 sendmail

Type: path
Default: "/usr/sbin/sendmail -oem -oi"

Specifies the program and arguments used to deliver mail sent by Mutt. Mutt expects that the specified program interprets additional arguments as recipient addresses.

### 6.3.238 sendmail_wait

Type: number
Default: 0

Specifies the number of seconds to wait for the "6.3.237 ($sendmail)" process to finish before giving up and putting delivery in the background.

Mutt interprets the value of this variable as follows:

**>0**

number of seconds to wait for sendmail to finish before continuing

**0**

wait forever for sendmail to finish

**<0**

always put sendmail in the background without waiting

Note that if you specify a value other than 0, the output of the child process will be put in a temporary file. If there is some error, you will be informed as to where to find the output.

### 6.3.239 shell

Type: path
Default: ""

Command to use when spawning a subshell. By default, the user's login shell from /etc/passwd is used.

#### 6.3.240 sig_dashes

Type: boolean
Default: yes

If set, a line containing "– " will be inserted before your "6.3.242 ($signature)". It is **strongly** recommended that you not unset this variable unless your "signature" contains just your name. The reason for this is because many software packages use "– \n" to detect your signature. For example, Mutt has the ability to highlight the signature in a different color in the builtin pager.

#### 6.3.241 sig_on_top

Type: boolean
Default: no

If set, the signature will be included before any quoted or forwarded text. It is **strongly** recommended that you do not set this variable unless you really know what you are doing, and are prepared to take some heat from netiquette guardians.

#### 6.3.242 signature

Type: path
Default: "~/.signature"

Specifies the filename of your signature, which is appended to all outgoing messages. If the filename ends with a pipe ("|"), it is assumed that filename is a shell command and input should be read from its stdout.

#### 6.3.243 simple_search

Type: string
Default: "~f %s | ~s %s"

Specifies how Mutt should expand a simple search into a real search pattern. A simple search is one that does not contain any of the ~ operators. See "4.2 (patterns)" for more information on search patterns.

For example, if you simply type joe at a search or limit prompt, Mutt will automatically expand it to the value specified by this variable. For the default value it would be:

~f joe | ~s joe

#### 6.3.244 smart_wrap

Type: boolean
Default: yes

Controls the display of lines longer than the screen width in the internal pager. If set, long lines are wrapped at a word boundary. If unset, lines are simply wrapped at the screen edge. Also see the "6.3.101 ($markers)" variable.

### 6.3.245 smileys

Type: regular expression
Default: "(>From )|(:[-^]?[][])(><}{|/DP])"

The *pager* uses this variable to catch some common false positives of "6.3.215 ($quote_regexp)", most notably smileys in the beginning of a line

### 6.3.246 sleep_time

Type: number
Default: 1

Specifies time, in seconds, to pause while displaying certain informational messages, while moving from folder to folder and after expunging messages from the current folder. The default is to pause one second, so a value of zero for this option suppresses the pause.

### 6.3.247 sort

Type: sort order
Default: date

Specifies how to sort messages in the *index* menu. Valid values are:

```
date or date-sent
date-received
from
mailbox-order (unsorted)
score
size
spam
subject
threads
to
```

You may optionally use the reverse- prefix to specify reverse sorting order (example: set sort=reverse-date-sent).

### 6.3.248 sort_alias

Type: sort order
Default: alias

Specifies how the entries in the 'alias' menu are sorted. The following are legal values:

```
address (sort alphabetically by email address)
alias (sort alphabetically by alias name)
unsorted (leave in order specified in .muttrc)
```

### 6.3.249 sort_aux

Type: sort order
Default: date

When sorting by threads, this variable controls how threads are sorted in relation to other threads, and how the branches of the thread trees are sorted. This can be set to any value that "6.3.247 ($sort)" can, except threads (in that case, mutt will just use date-sent). You can also specify the last- prefix in addition to the reverse- prefix, but last- must come after reverse-. The last- prefix causes messages to be sorted against its siblings by which has the last descendant, using the rest of sort_aux as an ordering. For instance, set sort_aux=last- date-received would mean that if a new message is received in a thread, that thread becomes the last one displayed (or the first, if you have set sort=reverse-threads.) Note: For reversed "6.3.247 ($sort)" order 6.3.249 ($sort_aux) is reversed again (which is not the right thing to do, but kept to not break any existing configuration setting).

### 6.3.250 sort_browser

Type: sort order
Default: alpha

Specifies how to sort entries in the file browser. By default, the entries are sorted alphabetically. Valid values:

```
alpha (alphabetically)
date
size
unsorted
```

You may optionally use the reverse- prefix to specify reverse sorting order (example: set sort_browser=reverse-date).

### 6.3.251 sort_re

Type: boolean
Default: yes

This variable is only useful when sorting by threads with "6.3.257 ($strict_threads)" unset. In that case, it changes the heuristic mutt uses to thread messages by subject. With sort_re set, mutt will only attach a message as the child of another message by subject if the subject of the child message starts with a substring matching the setting of "6.3.221 ($reply_regexp)". With sort_re unset, mutt will attach the message whether or not this is the case, as long as the non-"6.3.221 ($reply_regexp)" parts of both messages are identical.

### 6.3.252 spam_separator

Type: string
Default: ","

"6.3.252 (spam_separator)" controls what happens when multiple spam headers are matched: if unset, each successive header will overwrite any previous matches value for the spam label. If set, each successive match will append to the previous, using "6.3.252 (spam_separator)" as a separator.

#### 6.3.253 spoolfile

Type: path
Default: ""

If your spool mailbox is in a non-default place where Mutt cannot find it, you can specify its location with this variable. Mutt will automatically set this variable to the value of the environment variable $MAIL if it is not set.

#### 6.3.254 status_chars

Type: string
Default: "-*%A"

Controls the characters used by the "%r" indicator in "6.3.255 ($status_format)". The first character is used when the mailbox is unchanged. The second is used when the mailbox has been changed, and it needs to be resynchronized. The third is used if the mailbox is in read-only mode, or if the mailbox will not be written when exiting that mailbox (You can toggle whether to write changes to a mailbox with the toggle-write operation, bound by default to "%"). The fourth is used to indicate that the current folder has been opened in attach- message mode (Certain operations like composing a new mail, replying, forwarding, etc. are not permitted in this mode).

#### 6.3.255 status_format

Type: string
Default: "-%r-Mutt: %f [Msgs:%?M?%M/?%m%?n? New:%n?%?o? Old:%o?%?d? Del:%d?%?F? Flag:%F?%?t? Tag:%t?%?p? Post:%p?%?b? Inc:%b?%?l? %l?]—(%s/%S)-%>-(%P)—"

Controls the format of the status line displayed in the *index* menu. This string is similar to "6.3.92 ($index_format)", but has its own set of printf()-like sequences:

**%b**

number of mailboxes with new mail *

**%d**

number of deleted messages *

**%f**

the full pathname of the current mailbox

**%F**

number of flagged messages *

**%h**

local hostname

**%l**

size (in bytes) of the current mailbox *

**%L**

size (in bytes) of the messages shown (i.e., which match the current limit) *

**%m**

the number of messages in the mailbox *

**%M**

the number of messages shown (i.e., which match the current limit) *

**%n**

number of new messages in the mailbox *

**%o**

number of old unread messages *

**%p**

number of postponed messages *

**%P**

percentage of the way through the index

**%r**

modified/read-only/won't-write/attach-message indicator, according to 6.3.254 ($status_chars)

**%s**

current sorting mode (6.3.247 ($sort))

**%S**

current aux sorting method (6.3.249 ($sort_aux))

**%t**

number of tagged messages *

**%u**

number of unread messages *

**%v**

Mutt version string

**%V**

currently active limit pattern, if any *

**%>X**

right justify the rest of the string and pad with "X"

**%|X**

pad to the end of the line with "X"

* = can be optionally printed if nonzero

Some of the above sequences can be used to optionally print a string if their value is nonzero. For example, you may only want to see the number of flagged messages if such messages exist, since zero is not particularly meaningful. To optionally print a string based upon one of the above sequences, the following construct is used

%?<sequence_char>?<optional_string>?

where *sequence_char* is a character from the table above, and *optional_string* is the string you would like printed if *sequence_char* is nonzero. *optional_string* **may** contain other sequences as well as normal text, but you may **not** nest optional strings.

Here is an example illustrating how to optionally print the number of new messages in a mailbox: %?n?%n new messages.?

Additionally you can switch between two strings, the first one, if a value is zero, the second one, if the value is nonzero, by using the following construct: %?<sequence_char>?<if_string>&<else_string>?

You can additionally force the result of any printf-like sequence to be lowercase by prefixing the sequence character with an underscore (_) sign. For example, if you want to display the local hostname in lowercase, you would use: %_h

If you prefix the sequence character with a colon (:) character, mutt will replace any dots in the expansion by underscores. This might be helpful with IMAP folders that don't like dots in folder names.

#### 6.3.256 status_on_top

Type: boolean
Default: no

Setting this variable causes the "status bar" to be displayed on the first line of the screen rather than near the bottom.

#### 6.3.257 strict_threads

Type: boolean
Default: no

If set, threading will only make use of the "In-Reply-To" and "References" fields when you "6.3.247 ($sort)" by message threads. By default, messages with the same subject are grouped together in "pseudo threads." This may not always be desirable, such as in a personal mailbox where you might have several unrelated messages with the subject "hi" which will get grouped together.

#### 6.3.258 suspend

Type: boolean
Default: yes

When *unset*, mutt won't stop when the user presses the terminal's *susp* key, usually "control-Z". This is useful if you run mutt inside an xterm using a command like xterm -e mutt.

#### 6.3.259 text_flowed

Type: boolean
Default: no

When set, mutt will generate text/plain; format=flowed attachments. This format is easier to handle for some mailing software, and generally just looks like ordinary text. To actually make use of this format's features, you'll need support in your editor.

Note that 6.3.91 ($indent_string) is ignored when this option is set.

#### 6.3.260 thread_received

Type: boolean
Default: no

When set, mutt uses the date received rather than the date sent to thread messages by subject.

#### 6.3.261 thorough_search

Type: boolean
Default: no

Affects the ~*b* and ~*h* search operations described in section "4.2 (patterns)" above. If set, the headers and attachments of messages to be searched are decoded before searching. If unset, messages are searched as they appear in the folder.

#### 6.3.262 tilde

Type: boolean
Default: no

When set, the internal-pager will pad blank lines to the bottom of the screen with a tilde (~).

#### 6.3.263 timeout

Type: number
Default: 600

This variable controls the *number of seconds* Mutt will wait for a key to be pressed in the main menu before timing out and checking for new mail. A value of zero or less will cause Mutt to never time out.

#### 6.3.264 tmpdir

Type: path
Default: ""

This variable allows you to specify where Mutt will place its temporary files needed for displaying and composing messages. If this variable is not set, the environment variable TMPDIR is used. If TMPDIR is not set then "/tmp" is used.

#### 6.3.265 to_chars

Type: string
Default: " +TCFL"

Controls the character used to indicate mail addressed to you. The first character is the one used when the mail is NOT addressed to your address (default: space). The second is used when you are the only recipient of the message (default: +). The third is when your address appears in the TO header field, but you are not the only recipient of the message (default: T). The fourth character is used when your address is specified in the CC header field, but you are not the only recipient. The fifth character is used to indicate mail that was sent by *you*. The sixth character is used to indicate when a mail was sent to a mailing-list you subscribe to (default: L).

#### 6.3.266 tunnel

Type: string
Default: ""

Setting this variable will cause mutt to open a pipe to a command instead of a raw socket. You may be able to use this to set up preauthenticated connections to your IMAP/POP3 server. Example:

tunnel="ssh -q mailhost.net /usr/local/libexec/imapd"

NOTE: For this example to work you must be able to log in to the remote machine without having to enter a password.

#### 6.3.267 use_8bitmime

Type: boolean
Default: no

**Warning:** do not set this variable unless you are using a version of sendmail which supports the -B8BITMIME flag (such as sendmail 8.8.x) or you may not be able to send mail.

When *set*, Mutt will invoke "6.3.237 ($sendmail)" with the -B8BITMIME flag when sending 8-bit messages to enable ESMTP negotiation.

#### 6.3.268 use_domain

Type: boolean
Default: yes

When set, Mutt will qualify all local addresses (ones without the @host portion) with the value of "6.3.74 ($hostname)". If *unset*, no addresses will be qualified.

#### 6.3.269 use_from

Type: boolean
Default: yes

When *set*, Mutt will generate the 'From:' header field when sending messages. If *unset*, no 'From:' header field will be generated unless the user explicitly sets one using the "3.13 (my_hdr)" command.

#### 6.3.270 use_idn

Type: boolean
Default: yes

When *set*, Mutt will show you international domain names decoded. Note: You can use IDNs for addresses even if this is *unset*. This variable only affects decoding.

#### 6.3.271 use_ipv6

Type: boolean
Default: yes

When *set*, Mutt will look for IPv6 addresses of hosts it tries to contact. If this option is unset, Mutt will restrict itself to IPv4 addresses. Normally, the default should work.

#### 6.3.272 user_agent

Type: boolean
Default: yes

When *set*, mutt will add a "User-Agent" header to outgoing messages, indicating which version of mutt was used for composing them.

#### 6.3.273 visual

Type: path
Default: ""

Specifies the visual editor to invoke when the *~v* command is given in the builtin editor.

#### 6.3.274 wait_key

Type: boolean
Default: yes

Controls whether Mutt will ask you to press a key after *shell- escape*, *pipe-message*, *pipe-entry*, *print-message*, and *print-entry* commands.

It is also used when viewing attachments with "5.4 (auto_view)", provided that the corresponding mailcap entry has a *needsterminal* flag, and the external program is interactive.

When set, Mutt will always ask for a key. When unset, Mutt will wait for a key only if the external command returned a non-zero status.

#### 6.3.275 weed

Type: boolean
Default: yes

When set, mutt will weed headers when displaying, forwarding, printing, or replying to messages.

#### 6.3.276 wrap_search

Type: boolean
Default: yes

Controls whether searches wrap around the end of the mailbox.

When set, searches will wrap around the first (or last) message. When unset, searches will not wrap.

#### 6.3.277 wrapmargin

Type: number
Default: 0

Controls the size of the margin remaining at the right side of the terminal when mutt's pager does smart wrapping.

#### 6.3.278 write_inc

Type: number
Default: 10

When writing a mailbox, a message will be printed every *write_inc* messages to indicate progress. If set to 0, only a single message will be displayed before writing a mailbox.

Also see the "6.3.216 ($read_inc)" variable.

#### 6.3.279 write_bcc

Type: boolean
Default: yes

Controls whether mutt writes out the Bcc header when preparing messages to be sent. Exim users may wish to unset this.

### 6.4 Functions

The following is the list of available functions listed by the mapping in which they are available. The default key setting is given, and an explanation of what the function does. The key bindings of these functions can be changed with the 3.3 (bind) command.

### 6.4.1 generic

The *generic* menu is not a real menu, but specifies common functions (such as movement) available in all menus except for *pager* and *editor*. Changing settings for this menu will affect the default bindings for all menus (except as noted).

| | | |
|---|---|---|
| bottom-page | L | move to the bottom of the page |
| current-bottom | not bound | move current entry to bottom of page |
| current-middle | not bound | move current entry to middle of page |
| current-top | not bound | move current entry to top of page |
| enter-command | : | enter a muttrc command |
| exit | q | exit this menu |
| first-entry | = | move to the first entry |
| half-down | ] | scroll down 1/2 page |
| half-up | [ | scroll up 1/2 page |
| help | ? | this screen |
| jump | number | jump to an index number |
| last-entry | * | move to the last entry |
| middle-page | M | move to the middle of the page |
| next-entry | j | move to the next entry |
| next-line | > | scroll down one line |
| next-page | z | move to the next page |
| previous-entry | k | move to the previous entry |
| previous-line | < | scroll up one line |
| previous-page | Z | move to the previous page |
| refresh | ^L | clear and redraw the screen |
| search | / | search for a regular expression |
| search-next | n | search for next match |
| search-opposite | not bound | search for next match in opposite direction |
| search-reverse | ESC / | search backwards for a regular expression |
| select-entry | RET | select the current entry |
| shell-escape | ! | run a program in a subshell |
| tag-entry | t | toggle the tag on the current entry |
| tag-prefix | ; | apply next command to tagged entries |
| tag-prefix-cond | not bound | apply next function ONLY to tagged messages |
| top-page | H | move to the top of the page |
| what-key | not bound | display the keycode for a key press |

### 6.4.2 index

| | | |
|---|---|---|
| bounce-message | b | remail a message to another user |
| change-folder | c | open a different folder |
| change-folder-readonly | ESC c | open a different folder in read only mode |
| check-traditional-pgp | ESC P | check for classic pgp |
| clear-flag | W | clear a status flag from a message |
| copy-message | C | copy a message to a file/mailbox |
| create-alias | a | create an alias from a message sender |

```
decode-copy             ESC C   decode a message and copy it to a file/mailbox
decode-save             ESC s   decode a message and save it to a file/mailbox
delete-message              d   delete the current entry
delete-pattern              D   delete messages matching a pattern
delete-subthread        ESC d   delete all messages in subthread
delete-thread              ^D   delete all messages in thread
display-address             @   display full address of sender
display-toggle-weed         h   display message and toggle header weeding
display-message           RET   display a message
edit                        e   edit the current message
edit-type                  ^E   edit the current message's Content-Type
exit                        x   exit without saving changes
extract-keys               ^K   extract PGP public keys
fetch-mail                  G   retrieve mail from POP server
flag-message                F   toggle a message's 'important' flag
forget-passphrase          ^F   wipe PGP passphrase from memory
forward-message             f   forward a message with comments
group-reply                 g   reply to all recipients
limit                       l   show only messages matching a pattern
list-reply                  L   reply to specified mailing list
mail                        m   compose a new mail message
mail-key                ESC k   mail a PGP public key
next-new                  TAB   jump to the next new message
next-subthread          ESC n   jump to the next subthread
next-thread                ^N   jump to the next thread
next-undeleted              j   move to the next undeleted message
next-unread         not bound   jump to the next unread message
parent-message              P   jump to parent message in thread
pipe-message                |   pipe message/attachment to a shell command
previous-new          ESC TAB   jump to the previous new message
previous-page               Z   move to the previous page
previous-subthread      ESC p   jump to previous subthread
previous-thread            ^P   jump to previous thread
previous-undeleted          k   move to the last undelete message
previous-unread     not bound   jump to the previous unread message
print-message               p   print the current entry
query                       Q   query external program for addresses
quit                        q   save changes to mailbox and quit
read-subthread          ESC r   mark the current subthread as read
read-thread                ^R   mark the current thread as read
recall-message              R   recall a postponed message
reply                       r   reply to a message
resend-message          ESC e   resend message and preserve MIME structure
save-message                s   save message/attachment to a file
set-flag                    w   set a status flag on a message
show-version                V   show the Mutt version number and date
show-limit              ESC l   show currently active limit pattern, if any
```

| | | |
|---|---|---|
| sort-mailbox | o | sort messages |
| sort-reverse | O | sort messages in reverse order |
| sync-mailbox | $ | save changes to mailbox |
| tag-pattern | T | tag messages matching a pattern |
| tag-thread | ESC t | tag/untag all messages in the current thread |
| toggle-new | N | toggle a message's 'new' flag |
| toggle-write | % | toggle whether the mailbox will be rewritten |
| undelete-message | u | undelete the current entry |
| undelete-pattern | U | undelete messages matching a pattern |
| undelete-subthread | ESC u | undelete all messages in subthread |
| undelete-thread | ^U | undelete all messages in thread |
| untag-pattern | ^T | untag messages matching a pattern |
| view-attachments | v | show MIME attachments |

### 6.4.3 pager

| | | |
|---|---|---|
| bottom | not bound | jump to the bottom of the message |
| bounce-message | b | remail a message to another user |
| change-folder | c | open a different folder |
| change-folder-readonly | ESC c | open a different folder in read only mode |
| check-traditional-pgp | ESC P | check for classic pgp |
| copy-message | C | copy a message to a file/mailbox |
| create-alias | a | create an alias from a message sender |
| decode-copy | ESC C | decode a message and copy it to a file/mailbox |
| decode-save | ESC s | decode a message and save it to a file/mailbox |
| delete-message | d | delete the current entry |
| delete-subthread | ESC d | delete all messages in subthread |
| delete-thread | ^D | delete all messages in thread |
| display-address | @ | display full address of sender |
| display-toggle-weed | h | display message and toggle header weeding |
| edit | e | edit the current message |
| edit-type | ^E | edit the current message's Content-Type |
| enter-command | : | enter a muttrc command |
| exit | i | return to the main-menu |
| extract-keys | ^K | extract PGP public keys |
| flag-message | F | toggle a message's 'important' flag |
| forget-passphrase | ^F | wipe PGP passphrase from memory |
| forward-message | f | forward a message with comments |
| group-reply | g | reply to all recipients |
| half-up | not bound | move up one-half page |
| half-down | not bound | move down one-half page |
| help | ? | this screen |
| list-reply | L | reply to specified mailing list |
| mail | m | compose a new mail message |
| mail-key | ESC k | mail a PGP public key |
| mark-as-new | N | toggle a message's 'new' flag |

```
next-line                RET   scroll down one line
next-entry                 J   move to the next entry
next-new                 TAB   jump to the next new message
next-page                      move to the next page
next-subthread         ESC n   jump to the next subthread
next-thread              ^N    jump to the next thread
next-undeleted             j   move to the next undeleted message
next-unread        not bound   jump to the next unread message
parent-message             P   jump to parent message in thread
pipe-message               |   pipe message/attachment to a shell command
previous-line      BackSpace   scroll up one line
previous-entry             K   move to the previous entry
previous-new       not bound   jump to the previous new message
previous-page              -   move to the previous page
previous-subthread     ESC p   jump to previous subthread
previous-thread          ^P    jump to previous thread
previous-undeleted         k   move to the last undelete message
previous-unread    not bound   jump to the previous unread message
print-message              p   print the current entry
quit                       Q   save changes to mailbox and quit
read-subthread         ESC r   mark the current subthread as read
read-thread              ^R    mark the current thread as read
recall-message             R   recall a postponed message
redraw-screen            ^L    clear and redraw the screen
reply                      r   reply to a message
save-message               s   save message/attachment to a file
search                     /   search for a regular expression
search-next                n   search for next match
search-opposite    not bound   search for next match in opposite direction
search-reverse         ESC /   search backwards for a regular expression
search-toggle              \   toggle search pattern coloring
shell-escape               !   invoke a command in a subshell
show-version               V   show the Mutt version number and date
skip-quoted                S   skip beyond quoted text
sync-mailbox               $   save changes to mailbox
tag-message                t   tag a message
toggle-quoted              T   toggle display of quoted text
top                        ^   jump to the top of the message
undelete-message           u   undelete the current entry
undelete-subthread     ESC u   undelete all messages in subthread
undelete-thread          ^U    undelete all messages in thread
view-attachments           v   show MIME attachments
```

### 6.4.4 alias

```
search                     /   search for a regular expression
```

| | | |
|---|---|---|
| search-next | n | search for next match |
| search-reverse | ESC / | search backwards for a regular expression |

### 6.4.5 query

| | | |
|---|---|---|
| create-alias | a | create an alias from a message sender |
| mail | m | compose a new mail message |
| query | Q | query external program for addresses |
| query-append | A | append new query results to current results |
| search | / | search for a regular expression |
| search-next | n | search for next match |
| search-opposite | not bound | search for next match in opposite direction |
| search-reverse | ESC / | search backwards for a regular expression |

### 6.4.6 attach

| | | |
|---|---|---|
| bounce-message | b | remail a message to another user |
| collapse-parts | v | toggle display of subparts |
| delete-entry | d | delete the current entry |
| display-toggle-weed | h | display message and toggle header weeding |
| edit-type | ^E | edit the current entry's Content-Type |
| extract-keys | ^K | extract PGP public keys |
| forward-message | f | forward a message with comments |
| group-reply | g | reply to all recipients |
| list-reply | L | reply to specified mailing list |
| pipe-entry | \| | pipe message/attachment to a shell command |
| print-entry | p | print the current entry |
| reply | r | reply to a message |
| resend-message | ESC e | resend message and preserve MIME structure |
| save-entry | s | save message/attachment to a file |
| undelete-entry | u | undelete the current entry |
| view-attach | RET | view attachment using mailcap entry if necessary |
| view-mailcap | m | force viewing of attachment using mailcap |
| view-text | T | view attachment as text |

### 6.4.7 compose

| | | |
|---|---|---|
| attach-file | a | attach a file(s) to this message |
| attach-message | A | attach message(s) to this message |
| attach-key | ESC k | attach a PGP public key |
| copy-file | C | save message/attachment to a file |
| detach-file | D | delete the current entry |
| display-toggle-weed | h | display message and toggle header weeding |
| edit-bcc | b | edit the BCC list |
| edit-cc | c | edit the CC list |
| edit-description | d | edit attachment description |

| | | |
|---|---|---|
| edit-encoding | ^E | edit attachment transfer-encoding |
| edit-fcc | f | enter a file to save a copy of this message in |
| edit-from | ESC f | edit the from: field |
| edit-file | ^X e | edit the file to be attached |
| edit-headers | E | edit the message with headers |
| edit | e | edit the message |
| edit-mime | m | edit attachment using mailcap entry |
| edit-reply-to | r | edit the Reply-To field |
| edit-subject | s | edit the subject of this message |
| edit-to | t | edit the TO list |
| edit-type | ^T | edit attachment type |
| filter-entry | F | filter attachment through a shell command |
| forget-passphrase | ^F | wipe PGP passphrase from memory |
| ispell | i | run ispell on the message |
| new-mime | n | compose new attachment using mailcap entry |
| pgp-menu | p | show PGP options |
| pipe-entry | \| | pipe message/attachment to a shell command |
| postpone-message | P | save this message to send later |
| print-entry | l | print the current entry |
| rename-file | R | rename/move an attached file |
| send-message | y | send the message |
| toggle-unlink | u | toggle whether to delete file after sending it |
| view-attach | RET | view attachment using mailcap entry if necessary |
| write-fcc | w | write the message to a folder |

### 6.4.8 postpone

| | | |
|---|---|---|
| delete-entry | d | delete the current entry |
| undelete-entry | u | undelete the current entry |

### 6.4.9 browser

| | | |
|---|---|---|
| change-dir | c | change directories |
| check-new | TAB | check mailboxes for new mail |
| enter-mask | m | enter a file mask |
| search | / | search for a regular expression |
| search-next | n | search for next match |
| search-reverse | ESC / | search backwards for a regular expression |
| select-new | N | select a new file in this directory |
| sort | o | sort messages |
| sort-reverse | O | sort messages in reverse order |
| toggle-mailboxes | TAB | toggle whether to browse mailboxes or all files |
| view-file | SPACE | view file |
| subscribe | s | subscribe to current mailbox (IMAP Only) |
| unsubscribe | u | unsubscribe to current mailbox (IMAP Only) |
| toggle-subscribed | T | toggle view all/subscribed mailboxes (IMAP Only) |

### 6.4.10 pgp

| | | |
|---|---|---|
| view-name | % | view the key's user id |
| verify-key | c | verify a PGP public key |

### 6.4.11 editor

| | | |
|---|---|---|
| backspace | BackSpace | delete the char in front of the cursor |
| backward-char | ^B | move the cursor one character to the left |
| backward-word | ESC b | move the cursor to the previous word |
| bol | ^A | jump to the beginning of the line |
| buffy-cycle | Space | cycle among incoming mailboxes |
| capitalize-word | ESC c | uppercase the first character in the word |
| complete | TAB | complete filename or alias |
| complete-query | ^T | complete address with query |
| delete-char | ^D | delete the char under the cursor |
| downcase-word | ESC l | lowercase all characters in current word |
| eol | ^E | jump to the end of the line |
| forward-char | ^F | move the cursor one character to the right |
| forward-word | ESC f | move the cursor to the next word |
| history-down | not bound | scroll down through the history list |
| history-up | not bound | scroll up through the history list |
| kill-eol | ^K | delete chars from cursor to end of line |
| kill-eow | ESC d | delete chars from cursor to end of word |
| kill-line | ^U | delete all chars on the line |
| kill-word | ^W | delete the word in front of the cursor |
| quote-char | ^V | quote the next typed key |
| transpose-chars | not bound | transpose character under cursor with previous |
| upcase-word | ESC u | uppercase all characters in current word |

# 7 Miscellany

## 7.1 Acknowledgements

Kari Hurtta <*kari.hurtta@fmi.fi*>

co-developed the original MIME parsing code back in the ELM-ME days.

The following people have been very helpful to the development of Mutt:

Vikas Agnihotri <*vikasa@writeme.com*> ,
Francois Berjon <*Francois.Berjon@aar.alcatel-alsthom.fr*> ,
Aric Blumer <*aric@fore.com*> ,
John Capo <*jc@irbs.com*> ,
David Champion <*dgc@uchicago.edu* ,
Brendan Cully <*brendan@kublai.com*> ,
Liviu Daia <*daia@stoilow.imar.ro*> ,

Thomas E. Dickey <*dickey@herndon4.his.com*> ,
David DeSimone <*fox@convex.hp.com*> ,
Nickolay N. Dudorov <*nnd@wint.itfs.nsk.su*> ,
Ruslan Ermilov <*ru@freebsd.org*> ,
Edmund Grimley Evans <*edmundo@rano.org* ,
Michael Finken <*finken@conware.de*> ,
Sven Guckes <*guckes@math.fu-berlin.de*> ,
Lars Hecking <*lhecking@nmrc.ie*> ,
Mark Holloman <*holloman@nando.net*> ,
Andreas Holzmann <*holzmann@fmi.uni-passau.de*> ,
Marco d'Itri <*md@linux.it*> ,
Björn Jacke <*bjacke@suse.com*> ,
Byrial Jensen <*byrial@image.dk*> ,
David Jeske <*jeske@igcom.net*> ,
Christophe Kalt <*kalt@hugo.int-evry.fr*> ,
Tommi Komulainen <*Tommi.Komulainen@iki.fi*> ,
Felix von Leitner (a.k.a "Fefe") <*leitner@math.fu-berlin.de*> ,
Brandon Long <*blong@fiction.net*> ,
Jimmy Mäkelä <*jmy@flashback.net*> ,
Lars Marowsky-Bree <*lmb@pointer.in-minden.de*> ,
Thomas "Mike" Michlmayr <*mike@cosy.sbg.ac.at*> ,
Andrew W. Nosenko <*awn@bcs.zp.ua*> ,
David O'Brien <*obrien@Nuxi.cs.ucdavis.edu*> ,
Clint Olsen <*olsenc@ichips.intel.com*> ,
Park Myeong Seok <*pms@romance.kaist.ac.kr*> ,
Thomas Parmelan <*tom@ankh.fr.eu.org*> ,
Ollivier Robert <*roberto@keltia.freenix.fr*> ,
Thomas Roessler <*roessler@does-not-exist.org*> ,
Roland Rosenfeld <*roland@spinnaker.de*> ,
TAKIZAWA Takashi <*taki@luna.email.ne.jp*> ,
Allain Thivillon <*Allain.Thivillon@alma.fr*> ,
Gero Treuner <*gero@faveve.uni-stuttgart.de*> ,
Vsevolod Volkov <*vvv@lucky.net*> ,
Ken Weinert <*kenw@ihs.com*>

## 7.2 About this document

This document was written in SGML, and then rendered using the *sgml-tools* package.